2016

Probabilistic and Deep Learning Algorithms for the Analysis of Imagery Data

Saikat Basu
Louisiana State University and Agricultural and Mechanical College

PROBABILISTIC AND DEEP LEARNING ALGORITHMS FOR THE ANALYSIS OF IMAGERY DATA

A Dissertation

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Doctor Of Philosophy

in

The Department of Computer Science

by
Saikat Basu
B.Tech, National Institute of Technology Durgapur, 2011
December 2016

Dedicated to Baba and Maa without whom this would never have been possible.

Acknowledgments

This research was supported by NASA Carbon Monitoring System through Grant #NNH14Z DA001-N-CMS and Army Research Office (ARO) under Grant #W911NF1010495 and was partially supported by the Cooperative Agreement Number NASA-NNX12AD05A, CFDA Number 43.001, for the project identified as "Ames Research Center Cooperative for Research in Earth Science and Technology (ARC-CREST)". Any opinions findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect that of NASA, ARO or the United States Government. The research was also partially supported by the AWS Climate Research Grant. We are grateful to the United States Department of Agriculture for providing us the National Agriculture Imagery Program (NAIP) airborne imagery dataset for the Continental United States.

I would like to thank my advisor Supratik Mukhopadhyay for his constant help and guidance and for motivating me to pursue research in Deep Learning. I am also thankful to my committee members Costas Busch, Jianhua Chen, Hartmut Kaiser and Ye-Sho Chen for providing me valuable feedback and in reviewing my research work. I am also thankful to the members of my research group Robert DiBiano, Manohar Karki and Malcolm Stagg for their time and support.

I would also like to thank Sangram Ganguly, Ramakrishna Nemani, Andrew Michaelis, Petr Votava, Uttam Kumar, Cristina Milesi and other members from NASA Ames Research Center who helped and guided me during various phases of my research.

I am thankful to Ananya for putting up with me and Subhajit, Sayan, Arnab, Sujana, Ishita, Trina, Arghya, Satadru and Joy for those amazing weekends and the delicious food. I am also thankful to Ayan for the inspiring conversations. Finally, I would like to thank Baba, Maa and Didi for being there with me at every turn of my life and for being a constant source of motivaton.

Table of Contents

List of Tables

List of Figures

Abstract

Accurate object classification is a challenging problem for various low to high resolution imagery data. This applies to both natural as well as synthetic image datasets. However, each object recognition dataset poses its own distinct set of domain-specific problems. In order to address these issues, we need to devise intelligent learning algorithms which require a deep understanding and careful analysis of the feature space. In this thesis, we introduce three new learning frameworks for the analysis of both airborne images (NAIP dataset) and handwritten digit datasets without and with noise (MNIST and n-MNIST respectively).

First, we propose a probabilistic framework for the analysis of the NAIP dataset which includes (1) an unsupervised segmentation module based on the Statistical Region Merging algorithm, (2) a feature extraction module that extracts a set of standard hand-crafted texture features from the images, (3) a supervised classification algorithm based on Feedforward Backpropagation Neural Networks, and (4) a structured prediction framework using Conditional Random Fields that integrates the results of the segmentation and classification modules into a single composite model to generate the final class labels.

Next, we introduce two new datasets SAT-4 and SAT-6 sampled from the NAIP imagery and use them to evaluate a multitude of Deep Learning algorithms including Deep Belief Networks (DBN), Convolutional Neural Networks (CNN) and Stacked Autoencoders (SAE) for generating class labels. Finally, we propose a learning framework by integrating hand-crafted texture features with a DBN. A DBN uses an unsupervised pre-training phase to perform initialization of the parameters of a Feedforward Backpropagation Neural Network to a global error basin which can then be improved using a round of supervised fine-tuning using Feedforward Backpropagation Neural Networks. These networks can subsequently be used for classification. In the following discussion, we show that the integration of hand-crafted features with DBN shows significant improvement in performance as compared to traditional DBN models which take raw image pixels as input. We also investigate why this integration proves to be particularly useful for aerial datasets using a statistical analysis based on Distribution Separability Criterion.

Then we introduce a new dataset called noisy-MNIST (n-MNIST) by adding (1) additive white gaussian noise (AWGN), (2) motion blur and (3) Reduced contrast and AWGN to the MNIST dataset and present a learning algorithm by combining probabilistic quadtrees and Deep Belief Networks. This dynamic integration of the Deep Belief Network with the probabilistic quadtrees provide significant improvement over traditional DBN models on both the MNIST and the n-MNIST datasets.

Finally, we extend our experiments on aerial imagery to the class of general texture images and present a theoretical analysis of Deep Neural Networks applied to texture classification. We derive the size of the feature space of textural features and also derive the Vapnik-Chervonenkis dimension of certain classes of Neural Networks. We also derive some useful results on intrinsic dimension and relative contrast of texture datasets and use these to highlight the differences between texture datasets and general object recognition datasets.

Chapter 1
Introduction

With the acquisition of a multitude of imagery data, the need to detect objects of interest in these datasets has grown exponentially over the past few decades. These images may be acquired using a variety of sensors ranging from handheld mobile devices and cameras to airborne and satellite sensors. Object recognition is useful both for natural as well as synthetic image datasets. However, each object recognition dataset poses its own distinct set of domain-specific problems. For instance, the features learnt from object recognition datasets ([31], [105]) are different from the features learnt from aerial or satellite datasets ([108], [104]). In order to address these issues, we need to devise intelligent learning algorithms which require a deep understanding and careful analysis of the feature space. In this thesis, we focus on aerial imagery and handwriten digit datasets and introduce two new learning frameworks for the analysis of aerial images (NAIP dataset [108]) and one for the analysis of handwritten digit datasets without and with noise (MNIST [105] and n-MNIST [107] respectively).

We need Very High Resolution (VHR) landcover classification maps in order to increase the accuracy of various land ecosystem outputs as well as those generated from climate models. There are limited studies in the literature which showcase state-of-the-art results in deriving VHR products from land cover data. Additionally, most methods rely heavily on commercial software packages which are difficult to scale to continental and global scales given the area of study. Challenges in current approaches relate to (a) large scale image data processing, (b) high computational cost, (c) highly distributed/parallel architecture, and (d) efficient machine learning algorithms. VHR aerial/satellite datasets measure in terabytes and feature vectors extracted from them amount to petabytes of data. The following chapters demonstrate the use of two scalable machine learning algorithms - one using Feedforward Backpropagation Neural Networks and another using Deep Belief Networks on aerial imagery data acquired by the National Agricultural Imaging Program (NAIP) for the whole of Continental United States (CONUS) at a spatial

resolution of 1 m/pixel. This data comes in the form of image tiles (a total of 330,000 image scenes) that are multispectral in nature (Red, Green, Blue and Near-infrared spectral channels) and has a total size of 65 terabytes for an individual acquisition over CONUS. Then, we propose a learning framework based on probabilistic quadtrees and Deep Belief Nets for classifying noisy datasets. Finally, we present a theoretical analysis of Deep Neural Networks pertinent to texture classification.

The rest of the thesis is organized as follows:

- Chapter 2 showcases an end-to-end architecture for designing the learning algorithm using feature extraction, Feedforward Backpropagation Neural Networks for image classification, a Statistical Region Merging (SRM) based segmentation algorithm to perform unsupervised segmentation and a structured prediction framework using Conditional Random Field (CRF) that integrates the results of the classification module and the segmentation module to create per-pixel labels.

- Chapter 3 proposes a learning framework based on hand-crafted features and Deep Belief Networks (DBN) that performs a round of unsupervised pre-training which is used to initialize the parameters of a Feedforward Backpropagation Neural Network using supervised fine-tuning. We compare our framework with three state-of-the-art object recognition algorithms, namely - Deep Belief Networks, Convolutional Neural Networks and Stacked Autoencoders. For performance evaluation, we use two datasets SAT-4 and SAT-6 extracted from the NAIP dataset. SAT-4 contains 4 labeled landcover classes namely barren land, trees, grassland and a class that consists of all land cover classes other than the above three. SAT-6 consists of 6 landcover classes namely barren land, trees, grassland, roads, buildings and water bodies. On the SAT-4 dataset, our best network produces a classification accuracy of 97.95% and outperforms the other three object recognition algorithms by $\sim$11%. On SAT-6, it produces a classification accuracy of 93.9% and outperforms the other algorithms by $\sim$15%. Comparative studies with a Random Forest classifier show the advantage of an unsupervised learning approach over traditional supervised learning

techniques.

- Chapter 4 proposes a new learning framework by integrating Probabilistic Quadtrees and Deep Belief Nets. We introduce two new datasets called noisy-MNIST (n-MNIST) and noisy Bangla (n-Bangla) datasets based on the MNIST and Bangla handwritten digit datasets by inserting (1) Additive White Gaussian Noise, (2) Motion Blur and (3) by a combination of Additive Noise and Reduced Contrast. We subsequently evaluate the performance of our framework on the original MNIST and Bangla datasets as well as the proposed n-MNIST and n-Bangla datasets. On these datasets, our framework shows promising results and significantly outperforms traditional Deep Belief Networks.

- Chapter 5 investigates the use of Deep Neural Networks for the classification of image datasets where texture features are important for generating class-conditional discriminative representations. Here we derive the size of the feature space for some standard textural features extracted from the input dataset and then use the theory of Vapnik-Chervonenkis dimension to show that hand-crafted feature extraction creates low-dimensional representations which help in reducing the overall excess error rate. We also derive the upper bounds on the VC dimension of Convolutional Neural Network as well as Dropout and Dropconnect networks and the relation between excess error rate of Dropout and Dropconnect networks. The concept of *intrinsic dimension* is used to validate the intuition that texture-based datasets are inherently higher dimensional as compared to handwritten digits or other object recognition datasets and hence more difficult to be shattered by neural networks. We then derive the mean distance from the centroid to the nearest and farthest sampling points in an n-dimensional manifold and show that the *Relative Contrast* of the sample data vanishes as dimensionality of the underlying vector space tends to infinity.

- Chapter 6 summarizes the useful results of the thesis both in terms of theoretical contributions as well as experimental studies and enumerates possible future directions of research.

Chapter 2
A Probabilistic Framework for Tree Cover Delineation in Aerial Imagery[1][2]

Accurate tree cover estimates are useful to derive Above Ground Biomass (AGB) density estimates from Very High Resolution (VHR) satellite imagery data. Numerous algorithms have been designed to perform tree cover delineation in high to coarse resolution satellite imagery, but most of them do not scale to terabytes of data, typical in these VHR datasets. In this chapter, we present an automated probabilistic framework for the segmentation and classification of 1-m VHR data as obtained from the National Agriculture Imagery Program (NAIP) for deriving tree cover estimates for the whole of Continental United States, using a High Performance Computing Architecture. The results from the classification and segmentation algorithms are then consolidated into a structured prediction framework using a discriminative undirected probabilistic graphical model based on Conditional Random Field (CRF), which helps in capturing the higher order contextual dependencies between neighboring pixels. Once the final probability maps are generated, the framework is updated and re-trained by incorporating expert knowledge through the relabeling of misclassified image patches. This leads to a significant improvement in the true positive rates and reduction in false positive rates. The tree cover maps were generated for the state of California, which covers a total of 11,095 NAIP tiles and spans a total geographical area of 163,696 sq. miles. Our framework produced correct detection rates of around 88% for fragmented forests and 74% for urban tree cover areas, with false positive rates lower than 2% for both regions. Comparative studies with the National Land Cover Data (NLCD) algorithm and the LiDAR high-resolution canopy height model showed the effectiveness of our algorithm for generating accurate high-resolution tree-cover maps.

[1] Adapted from the paper tiltled "A Semiautomated Probabilistic Framework for Tree-Cover Delineation From 1-m NAIP Imagery Using a High-Performance Computing Architecture" published in IEEE Transactions on Geoscience and Remote Sensing [10].

[2] Note that the terms aerial imagery and satellite imagery are used interchangeably throughout this thesis because the extracted features and learning algorithms are generic enough to handle both satellite and aerial imagery datasets.

2.1 Introduction

An unsolved problem with low to high resolution satellite-derived forest cover maps is their inaccuracy, particularly over heterogeneous landscapes, and the high degree of uncertainty they introduce when they are used for forest carbon mapping applications. Previous efforts have acknowledged the issues pertaining to misclassification errors in coarser resolution satellite-derived land cover products, however, limited studies are in place that demonstrate how very high resolution (VHR) land cover products at 1-m spatial resolution could improve regional estimations of Above Ground Biomass (AGB). This chapter develops techniques and algorithms designed to improve the accuracy of current satellite-based AGB maps as well as provide a reference layer for more accurately estimating regional AGB densities from the Forest Inventory and Analysis (FIA). The VHR tree-cover map can be used to compute tree-cover estimates at any low to high resolution spatial grid, reducing the uncertainties in estimating AGB density and mitigating the present shortcomings of medium-to-coarse resolution land-cover maps.

The principal challenges in computing VHR estimates of tree cover at 1-m are associated with (a) the high variability in land cover types as recognizable from satellite imagery, (b) data quality affected by conditions during acquisition and pre-processing, and (c) corruption of data due to atmospheric contamination and associated filtering techniques. Land cover class identification is difficult even through visual interpretation owing to high variance in atmospheric and lighting conditions, and manual delineation of tree cover from millions of imagery acquisitions is neither feasible nor cost-effective. Tree cover delineation can be mapped to an object recognition problem ([3], [91], [15], [43], [51]), which can be framed in two ways: a boundary delineation problem that can be solved by perceptual grouping or a bounding box extraction problem that is addressed using a classification framework that performs a binary/multi-class classification on the bounding box. Perceptual grouping employs a segmentation module that clusters contextually related objects/object parts into a single unified region ([72],[88],[89],[94]). In [72], the authors present a framework to cluster contextually related pixels into buildings from aerial imagery data. In [88], the authors propose a segmentation algorithm using multi-scale intensity analysis on nat-

ural images. In [89], the authors present an algorithm for perceptual grouping by using global image descriptors. They model the image as a graph and find an efficient image segmentation using normalized cuts. In [94], the authors present a Bayesian framework for segmentating images, which models the scenes in natural images as a parse graph using reversible Markov chains. On the other hand, a classification framework uses a variety of learning algorithms, such as boosting ([78],[70]), random forests ([25],[40],[39]), Support Vector Machines [4] and various others for performing both supervised and unsupervised classification of image patches based on visual and spectral characteristics. Neural Network and Deep Learning based approaches for tree cover delineation has been employed in various works in the literature ([8], [38], [115], [7] and [37]). Our work combines both the object classification and segmentation based approaches into a unified framework that performs a classification for individual pixels using feature descriptors extracted from a neighborhood (defined on a window centered at the pixel of interest) and then performs a perceptual grouping of pixels sharing similar visual and spectral signatures.

Present classification algorithms used for Moderate-resolution Imaging Spectroradiometer (MODIS) [35] or Landsat-based land cover maps, such as National Land Cover Data (NLCD) [100], produce accuracies of 75% and 78%, respectively. The MODIS algorithm works on 500-m resolution imagery; the NLCD works at 30-m resolution. The accuracy assessment is performed on a per-pixel basis and the relatively lower resolution of the dataset makes it difficult to analyze the performance of these algorithms for 1-m imagery. Thus, there is a pressing need for creating high resolution forest cover maps at a resolution of 1 m to improve accuracy in land cover maps and to improve several prognostic and diagnostic models that require land cover maps as input. An automated approach for tree crown classification was proposed in [82], based on the identification of tree apexes and the maximum rate of change in spectral reflectance along transect extending outward from the tree center. The algorithm was applied to sub-meter resolution imagery (at most up to 30 cm) but its accuracy decreased consistently and non-linearly with increasing pixel spacing or decreasing sampling resolution. Other approaches for tree crown classification based on the distribution of pixel brightness are proposed in [60] and [66]. [60]

proposed evaluating the brightness distribution within the radius of a circle centered on each tree, with values near the center of the tree crown being larger than at the edges showing a test for a 150m by 150m IKONOS image. [66] applies a similar concept with the valley forming approach of [41], which treats variation in brightness in the imagery as topography, where bright pixels are peaks (the crowns) interspersed by valleys (the darker inter-tree spaces). Also here results are reported for a small test-area of 620×550 meter and hence it is unknown how the algorithm would perform on a larger test area with higher variability. Other novel classification algorithms based on Deep Neural Networks have been used in ([76],[96]). The framework in [76] is used for the recognition of roads in aerial images. Detecting trees is a much harder problem considering the significantly higher variability in tree-cover – trees can have various color and texture characteristics while roads have little variation in color or texture and belong to a fixed set of classes, such as concrete, mud, gravel, sand, etc. Another important feature in road detection is the incorporation of contextual information that improves accuracy of the classifier. On the other hand, a tree can be present beside another tree, a road, a building or even a water body. Thus, incorporating inter-class contextual information into our framework does not lead to significant improvements of the classification. [76] use a 64x64 detection window, which is a very large context for a tree-delineation problem in which an image patch might contain multiple classes, such as bare ground, roads, rooftops etc. and hence not suitable for the tree-classification problem. A method based on object detection using a Bayes framework and a subsequent clustering of the objects into a hierarchical model using Latent Dirichlet Allocation was proposed by [96], but accurate delineation of tree-cover areas demands the use of a different approach because of the need for higher accuracy and lack of useful contextual information (for e.g., detecting a parking lot can use the presence of multiple cars and their orientation as a useful feature, but, a tree-delineation problem lacks the presence of such contextual information encoded in neighboring objects of interest). Classification and/or Segmentation of 1-m or sub-meter resolution imagery is possible with commercial packages (ENVI, PCI Geomatica, etc.), but these tools are not scalable across millions of scenes in an automated manner. The algorithm proposed by [74] is similar to our ap-

proach, which uses a segmentation module and a Random Forest based classification module to assess tree cover in the National Agriculture Imagery Program (NAIP) data [108]. The algorithm demonstrates a viable operational tool for the classification of 1-m NAIP imagery and produces an overall accuracy of 84.8%. However, the analysis is based on the software Definiens Developer Professional [1], which affects the scalability and cost-effectiveness of the implementation to terabytes of data. Additionally, the authors limited the testing of the methodology to Pembina County in North Dakota, which covers an area of only 1,122 sq. miles as opposed to the 163,696 sq. miles in our implementation.

In this chapter, we present an automated probabilistic framework for the segmentation and classification of 1-m VHR NAIP data to derive accurate large-scale estimates of tree cover. The results generated by the segmentation and classification algorithms are consolidated using a discriminative undirected probabilistic graphical model that performs structured prediction and helps in capturing the higher order contextual dependencies between neighboring pixels. A detailed description of the NAIP dataset is given in Section 2.2. A comprehensive summary of the proposed framework and the High Performance Computing (HPC) implementation details are provided in Section 2.3. Section 2.4 discusses the results and performance analysis for our pilot demonstration of the algorithm over California.

2.2 Dataset

The NAIP dataset consists of a total of 330,000 scenes spanning the whole of the Continental United States (CONUS). We used the uncompressed Digital Ortho Quarter Quad tiles (DOQQs) which are GeoTIFF images with an area corresponding to the United States Geological Survey (USGS) topographic quadrangles. The average image tiles are $\sim$6000 pixels in width and $\sim$7000 pixels in height, and are approximately 200 megabytes each. The entire NAIP dataset for the Continental Unites State is $\sim$65 terabytes. The imagery was acquired at a 1-m ground sample distance (GSD) with a horizontal accuracy that lies within six meters of photo-identifiable ground control points [106]. The images consist of 4 bands – red, green, blue and Near Infrared (NIR). We performed the preliminary test of our algorithm and obtained tree-cover maps for the entire

state of California, a total of 11,095 image tiles in the NAIP dataset. Figure 2.1 shows some sample image patches from the NAIP dataset containing tree and non-tree areas.

The tree cover maps generated by our algorithm were validated against two high-resolution airborne LiDAR data footprints. The first set of LiDAR data (henceforth referred to as Area 1) was collected in the western Sierra Nevada mountain range over the Teakettle Experimental Forest in California. The LiDAR was flown in Summer, 2008 with the OPTECH GEMINI ALSM unit from University of Florida, with operation frequency of 100-125 kHz. The maximum scanning angle was 25°. Data was captured from an altitude of 600-750 m. The swath overlap was 50%-75% which yields an average density of return which is approximately 18 pts/m^2. LiDAR processing was conducted at the University of Maryland following [33]. A Digital Elevation Model (DEM) was fit to the lowest returns from the raw LiDAR returns, and smoothed to represent local topography. The value of each raw LiDAR return was reduced by the elevation value of the corresponding DEM pixel. A Canopy Height Model (CHM) was obtained at each pixel using the maximum LiDAR height at a resolution of 0.5 m per pixel. For the purpose of validation, the resolution of the LiDAR dataset was enhanced to 1 m per pixel.

The second set of LiDAR data (henceforth referred to as Area 2) was obtained in the Chester area in California, using the LiDAR, Hyperspectral and Thermal (G-LiHT) Airborne Imaging device from NASA Goddard [29]. NASA's Cessna 206 was used for acquiring the G-LiHT data. The spatial resolution of the final LiDAR data was 1 m.

2.3 Methodology

We have designed and implemented a scalable semi-automated probabilistic framework for the classification and segmentation of millions of scenes using a HPC architecture. The framework is robust to variability in land cover data as well as atmospheric and lighting conditions. Our framework consists of the following modules: (1) Unsupervised Segmentation, (2) Feature Extraction, (3) Supervised Classification, and (4) Per-pixel Labeling.

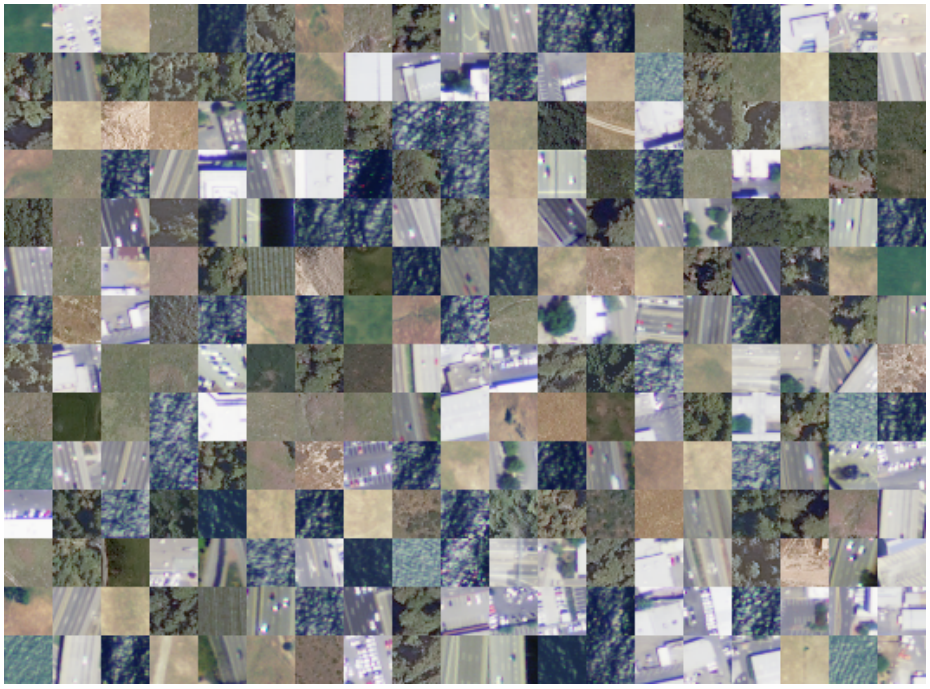


Figure 2.1: A sample of image patches from the NAIP dataset showing tree and non-tree areas.

2.3.1 Unsupervised Segmentation

We can define a segment as a cluster of image pixels having uniform values for the various spectral bands. The aim of segmentation is to find regions with uniform spectral chracteristics representing a particular land cover class. Segmentation is performed using the Statistical Region Merging (SRM) Algorithm [79]. We use a generalized SRM algorithm that incorporates values from all four bands. The SRM algorithm initially considers each pixel as a region and merges them to form larger regions based on a merging criterion. The merging criterion that we use in this case is as follows: Given the differences in red, green, blue and NIR values of neighboring pixels that correspond to dR, dG, dB and dNIR, respectively, merge two regions if (dR$<$threshold & dG$<$threshold & dB$<$threshold & dNIR$<$threshold). The merging criterion can be formalized as a merging predicate that is evaluated as true if two regions are merged and false otherwise. The generalized version of the merging predicate (adopted from [79]) can be formally written as

follows:

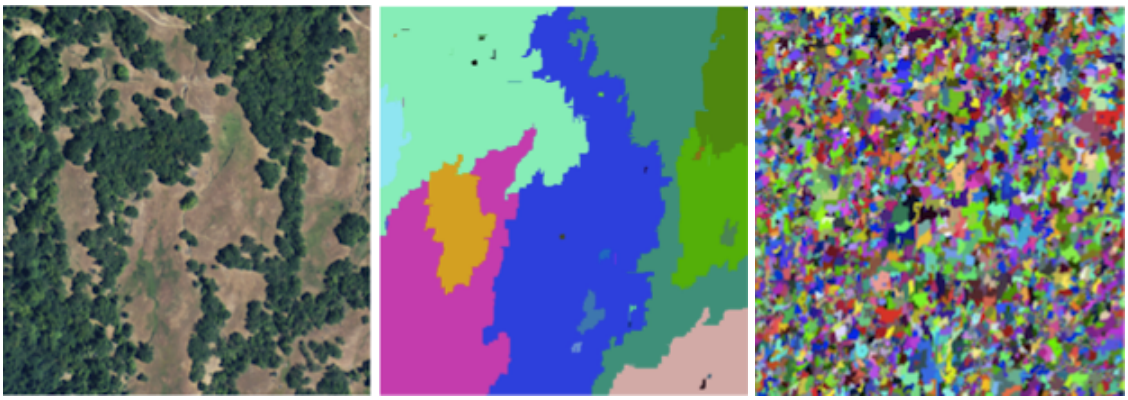
$$P(S,S') = \begin{cases} true, & \text{if } \forall c \in \{R,G,B,NIR\} \\ & |\bar{S}'_c - \bar{S}_c| \leq \sqrt{b^2(S) + b^2(S')} \\ false & \text{otherwise.} \end{cases} \tag{2.1}$$

where $\bar{S}_c$ and $\bar{S'}_c$ denote the mean value of the color channel c for regions S and S' respectively. b is a function defined as follows:

$$b(S) = g\sqrt{\frac{1}{2Q|S|} ln\left(\frac{|S_{|S|}|}{\delta}\right)} \tag{2.2}$$

where g is the number of possible values for each color channel (256 in our case). $|S|$ denotes the cardinality of a segment, i.e., the number of pixels within the boundaries of an image region S. $S_{|S|}$ represents the set of all regions that have the same cardinality as S. δ is a parameter that is inversely proportional to the image size. Q is the quantization parameter which is used to adjust the coarseness of the segmentation. A careful analysis of Equation 2.1 and Equation 2.2 shows that a higher value of Q results in a lower threshold thereby reducing the probability of two segments getting merged into a bigger segment, thus giving a finer segmentation. A lower value of Q results in a higher threshold and a coarser segmentation. The algorithm calculates the differences between neighboring pixels and sorts the pairs using radix sort. If the merging criterion is met, then it merges corresponding segments into one. We set a low threshold (or a higher Q value of 2^{15}) in order to perform over-segmentation. Each class (e.g. forest, grass, etc.) might be divided into multiple segments, but one segment would ideally not contain more than one class. This is useful for eliminating the possibility of inter-class overlap within a segment. Figure 3.2 shows an under-segmented and an over-segmented image. As can be seen in the under-segmented version, the same segment may contain both vegetated and non-vegetated areas.

In the case of an over-segmented image, areas within large homogeneous patches of vegetated pixels are split into multiple segments in the presence of spectral variability induced by



(a) Input Image (b) Input Image (c) Input Image

Figure 2.2: Example NAIP input image and under-segmented and over-segmented outputs from the Statistical Region Merging Algorithm. Under-segmentation creates interclass overlaps within a segment while over-segmentation avoids inter-class overlaps within a segment by creating mutually exclusive segments with pixels belonging to only one class.

factors such as shadows cast by tree/non-tree regions or the presence of dry brown patches within grassy areas, improving overall classification accuracy. SRM is more efficient compared to other segmentation algorithms, for example watershed and k-means clustering [26]. The lists of merging tests can be sorted using radix sort with color difference as the keys and hence has a time complexity of $O(|I|\log(g))$ which is linear in $|I|$. Here, $|I|$ is the cardinality or size of the input image. SRM segments a 512×512 image in about one second on an Intel Pentium 4 2.4G processor and hence is well suited for the current application involving terabytes of data. However, SRM has high memory requirements, around 3 Gigabytes per 6000×7000 image. This is mitigated by splitting the input image into 256×256 windows. This architectural implementation is detailed in Section 2.3.7.

2.3.2 Feature Extraction

Prior to the classification process, the feature extraction phase computes 150 features from the input imagery. The key features are mean, standard deviation, variance, 2nd moment, direct cosine transforms, correlation, co-variance, autocorrelation, energy, entropy, homogeneity, contrast, maximum probability and sum of variance of the hue, saturation, intensity, and NIR channels as well as those of the color co-occurrence matrices. These features were shown to

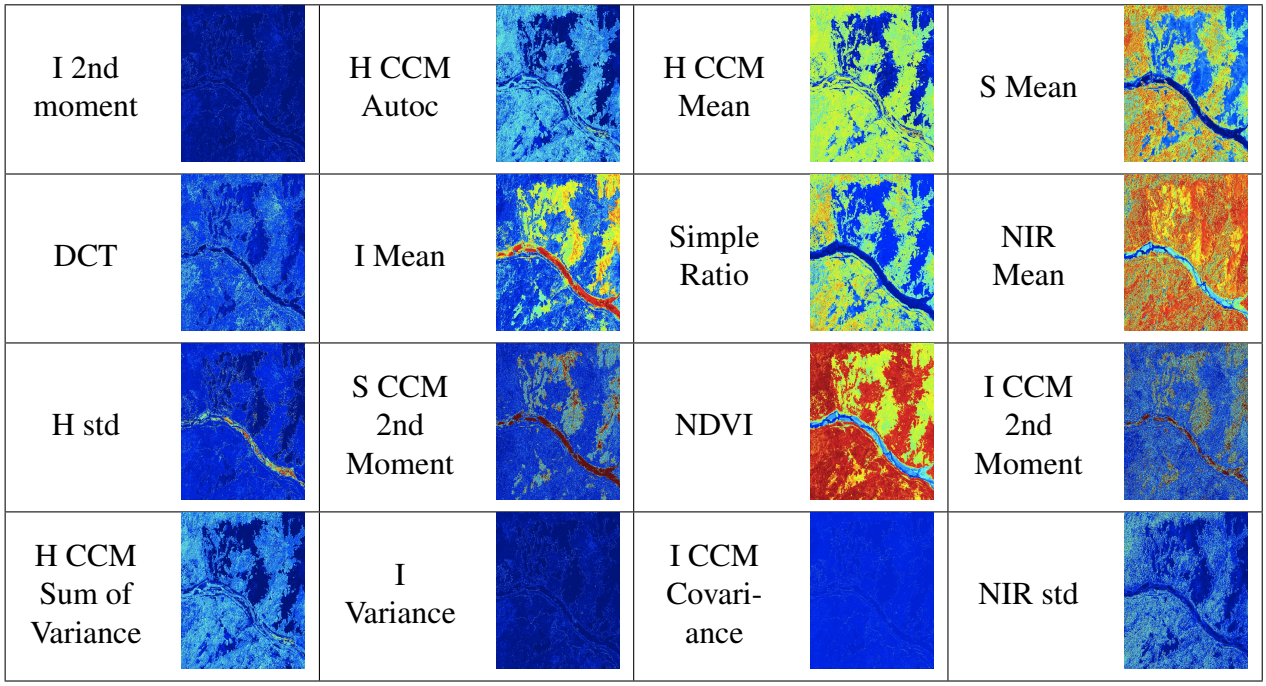


Figure 2.3: Example Features extracted from a sample NAIP tile.

be useful descriptors for classification of satellite imagery in previous studies ([46],[57],[28]). The Red band already provides a useful feature for delineating forests and non-forests based on chlorophyll reflectance, however, we also use derived features (vegetation indices derived from spectral band combinations) that are more representative of vegetation greenness, such as the Enhanced Vegetation Index (EVI) [54], Normalized Difference Vegetation Index (NDVI) ([85],[95]) and Atmospherically Resistant Vegetation Index (ARVI)[56].

These indices are expressed as :

$$EVI = G \times \frac{NIR - Red}{NIR + c_{red} \times Red - c_{blue} \times Blue + L} \tag{2.3}$$

Here, the coefficients G, c_{red}, c_{blue} and L are chosen to be 2.5, 6, 7.5 and 1, following those adopted in the MODIS EVI algorithm [106].

$$NDVI = \frac{NIR - Red}{NIR + Red} \tag{2.4}$$

$$ARVI = \frac{NIR - (2 \times Red - Blue)}{NIR + (2 \times Red + Blue)} \tag{2.5}$$

The performance of our machine learning-based approach depends to a large extent on the selected features. Some features contribute more than others towards optimal classification. The 150 features extracted are narrowed down to 22 using a feature-ranking algorithm based on Distribution Separability Criterion [19]. Some example image features are shown in Figure 2.3.

Sample Dataset	**Dist. between Means**	**Standard Deviations**
Raw Images	0.2163	0.1337
Extracted Features	**0.6712**	**0.0751**

Table 2.1: Distance between Means and Standard Deviations for raw image values and the Extracted feature vectors for a sample set of 5000 randomly selected labeled image patches from the NAIP dataset for the state of California.

- **Feature Ranking**

Improving classification accuracy can be viewed as maximizing the separability between the class-conditional distributions. Following the analysis presented in [19], we can view the problem of maximizing distribution separability as maximizing the distance between distribution means and minimizing their standard deviations. To quantify the properties of the NAIP dataset and the properties of the underlying distribution and to compare them to those of the extracted feature vectors, we randomly selected 5000 image patches from the NAIP tiles from the state of California and manually labeled as tree/non-tree. The labeling was done in an unbiased way, i.e., $\sim 50\%$ of the samples are chosen from tree samples and likewise for non-tree samples. Then we measured the distance between the mean values of the class conditional distributions and the standard deviations for both the raw pixel values as well as the features extracted in our framework. As illustrated in Table 3.6, the extracted features have a higher distance between means and a lower standard deviation as compared to the original image distributions, thereby ensuring better class separability. We can derive a metric for the Distribution Separability Criterion as follows:

$$D_s = \frac{\overline{\|\delta_{mean}\|}}{\overline{\delta_\sigma}} \tag{2.6}$$

where $\overline{\|\delta_{mean}\|}$ indicates the mean of distance between means and $\overline{\delta_\sigma}$ indicates the mean of standard deviations of the class conditional distributions. Maximizing D_s over the feature space, a feature ranking can be obtained. Table 3.7 shows the ranking of the various features used in our framework along with the values of the corresponding distance between means $\overline{\|\delta_{mean}\|}$, standard deviation $\overline{\delta_\sigma}$ and Distribution Separability Criterion D_s.

Rank	**Feature**	$\overline{\|\boldsymbol{\delta_{mean}}\|}$	$\overline{\boldsymbol{\delta_\sigma}}$	$\boldsymbol{D_s}$
1	I CCM mean	0.4031	0.1371	2.9403
2	H CCM sosvh	0.2359	0.0928	2.5413
3	H CCM autoc	0.2334	0.1090	2.1417
4	S CCM mean	0.0952	0.0675	1.4099
5	H CCM mean	0.0629	0.0560	1.1237
6	SR	0.0403	0.0428	0.9424
7	S CCM 2nd moment	0.0260	0.0312	0.8354
8	I CCM 2nd moment	0.0260	0.0312	0.8354
9	I 2nd moment	0.0260	0.0312	0.8345
10	I variance	0.0260	0.0312	0.8345
11	NIR std	0.0251	0.0315	0.7980
12	I std	0.0251	0.0314	0.7968
13	H std	0.0252	0.0317	0.7956
14	H mean	0.0240	0.0314	0.7632
15	I mean	0.0254	0.0336	0.7541
16	S mean	0.0232	0.0319	0.7268
17	I CCM covariance	0.0378	0.0522	0.7228
18	NIR mean	0.0246	0.0351	0.6997
19	ARVI	0.0229	0.0345	0.6622
20	NDVI	0.0215	0.0326	0.6594
21	DCT	0.0344	0.0594	0.5792
22	EVI	0.0144	0.0450	0.3207

Table 2.2: Ranking of features based on Distribution Separability Criterion for the sample dataset.

2.3.3 Classification

Classification is performed for each image pixel using feature descriptors defined on its neighborhood. A neighborhood system for a pixel p is a set $\prod_p$ defined as

$$\prod_p = \bigcup_{r_L - p_L \leq \tau} r \tag{2.7}$$

Here, r_L and p_L are the locations i.e., the ordered tuple (x, y) for the pixels r and p respectively, where, x and y are the X-coordinate (along the horizontal axis) and y is the Y-coordinates (along the vertical axis) respectively.

$$r_L - p_L = \delta \text{ if } r_L \text{ lies on a } \delta \times \delta \text{ window centered at } p_L \tag{2.8}$$

The neighborhood system for the pixel p_L is shown in Figure 2.4.

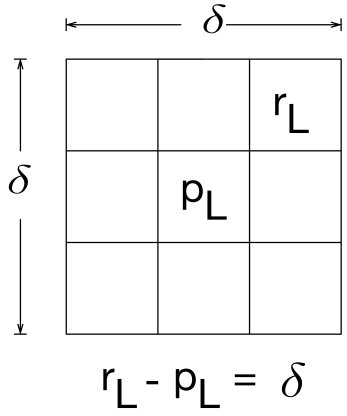


Figure 2.4: The neighborhood system for the pixel p_L where $r_L - p_L = \delta$

τ is the parameter that controls the extent of the neighborhood. τ was chosen to be 4 by experiment and a Receiver Operating Characteristic (ROC) Curve analysis as detailed in the results section. The ROC Curve analysis was used to select the optimal value for the parameter τ that resulted in the highest True Positive Rates for the detection window.

Our classification module consists of a probabilistic Neural Network framework that generates the posterior probability maps of the tree-cover estimates in the imagery data. The proposed

network consists of a fully connected backpropagation feed-forward Neural Network. In order to choose the optimal network architecture for the Neural Network classifier, we experimented with various network configurations along with the full set of 150 features as well as the set of 22 features selected in the feature selection stage highlighted in Section 2.3.2. The results are reported in Table 2.3. We observe from the table that the networks with 3 hidden layers produce lower classification accuracy than the networks with 2 hidden layers. This can be attributed to the limited amount of labeled training samples as compared to the model complexity of the deeper architectures which keeps increasing with depth of the network (for instance, 22700 free parameters for the 100-100-100 neural network) and hence resulting in over-fitting. In order to prevent over-fitting of the deeper networks, we employ two techniques - 1) L_2-norm regularization [47] and 2) Dropout [49]. For L_2 regularization we used a weight decay penalty of 10^{-4} and for the Dropout, we used a dropout fraction of 0.5. The results of the validation error of the 100-100 and 100-100-100 neural networks with varying epochs of the learning algorithm are presented in Figure 2.18. It can be seen that the 100-100-100 network with L_2 norm regularization and Dropout perform on an equal scale and produce lower validation error than the non-regularized version. However, it is interesting to see that the smaller non-regularized network with 2 hidden layers and 100 units in each layer outperforms the network with 3 hidden layers and 100 units in each layer even with regularization. So, it can be concluded from the experiments that both Dropout and L_2 norm regularization can act equally well as regularization techniques for Deeper Neural Networks, however, for a limited number of training samples, the shallow network with 2 hidden layers still produces lower classification errors on the held-out validation set. Following the results from Table 2.3, the best network was chosen as one with 2 hidden layers with 50 units in each and one output layer with one unit. The activation function is tanhyperbolic (tansigmoid) for hidden layers and linear for output layer:

$$\sigma(t) = \tanh(t) = \frac{e^t - e^{-t}}{e^t + e^{-t}} \tag{2.9}$$

The parameters are initialized using the Nguyen-Widrow Randomization algorithm [78]. The

mean squared error (MSE) is used as the performance metric. In the training phase we choose $\sim$100,000 sample points from each class. They are chosen randomly from various scenes ranging from urban landscapes to densely forested areas. An automated image labeling tool based on interactive segmentation developed as part of this study displays images randomly to a human expert who then labels the image patches as a tree cover or non-tree area, which are in turn saved to the training database along with proper labeling. Details of the image labeling tool are provided in Section 2.3.6.

The neural network gives an estimate of the posterior probabilities of the class labels, given the input vectors - the input vectors are the feature vectors extracted from the input image. As illustrated in [18], the outputs of a neural network trained by minimizing the mean squared error function approximates the conditional averages of the target data as

$$y_k(x) = \langle t_k|x\rangle = \int t_k p(t_k|x)dt_k \tag{2.10}$$

where t_k are the set of target values that represent the class membership of the input vector x_k and $p(t_k|x)$ is the probability that the input vector x attains the target value t_k. Thus, dt_k defines the differential over all target values t_k. To map the outputs of the neural network to the posterior probabilities of the labeling, we use a single output y and a target coding that sets $t^n = 1$ if x^n is from class C_1 and $t^n = 0$ if x^n is from class C_2. The target distribution can then be represented as

$$p(t_k|x) = \delta(t-1)P(C_1|x) + \delta(t)P(C_2|x) \tag{2.11}$$

Here, δ represents the Dirac delta function. This function exhibits the property $\delta(x) = 0$ if $x \neq 0$ and

$$\int_{-\infty}^{\infty} \delta(x)dx = 1 \tag{2.12}$$

From (7) and (8), we get

$$y(x) = P(C_1|x) \tag{2.13}$$

The network output $y(x)$ represents the posterior probability of the input vector x having the class membership C_1 and the probability of the class membership C_2 is given by $P(C_2|x) = 1 - y(x)$.

2.3.4 Conditional Random Field

A Conditional Random Field (CRF)[62] has been used in the pattern recognition literature for performing structured prediction [5]. In structured prediction the labeling we assign to a particular pixel is dependent on the feature values assumed by the pixel under consideration and also on the pixel values of "neighboring" pixels. The word "neighboring" here can either mean a 4-connected or 8-connected neighborhood or some custom metric defining the notion of neighborhood. The concept of neighborhood is useful in encoding contextual information. The final labeling of a pixel as a vegetated pixel depends not only on whether that pixel is classified as a tree, but also on the classification of neighboring pixels. For example, if a pixel has been classified as a tree pixel by the classifier and all the neighboring pixels have been classified as non-tree pixels, then, it is safe to assume with a high probability that the result of the classifier is due to random classification noise. A Conditional Random Field (CRF) is a kind of probabilistic graphical model which provides an encoding of contextual information using an undirected graph [62], [32]. The probability distributions are defined using a random variable X over a set of observations and another random variable L over an analogous set of label sequences. The index of L is denoted by the vertices or nodes of an undirected graph denoted by $G = (V, E)$ such that $L = (L_v)_{v \in V}$. The tuple (X, L) is known as a Conditional Random Field if the random variable L conditioned on the random variable X exhibits the Markov property in terms of the graphical model, i.e., $p(L_v|X, L_w, w \neq v) = p(L_v|X, L_w, w{\sim}v)$, where, $w{\sim}v$ denotes that the vertices w and v are neighboring vertices in G. Following the conventions defined in [58], the random vari-

able X is defined over a neighborhood system N and a lattice defined over a set $V = 1, 2, ..., n$. It is to be noted that this neighborhood system N should not be confused with the neighborhood system $\prod_p$ defined in Equation 2.7. $\prod_p$ indicates the neighborhood for the classification algorithm that determines the bounds of the decision boundaries for the classifier outcome for a particular pixel p, whereas, N denotes the system characterized by uniform probability distributions owing to similar visual and spectral characteristics, which takes the form of a segment in this case. In a CRF, a set of mutually conditionally dependent random variables X_C are enoceded in the form of a clique C. A probability distribution associated with any random variable X_i of a clique is conditionally dependent on the distributions of all other random variables in the clique. The objective function we use takes the form

$$E(x) = \sum_{p \in V} \phi_p(x_p) + \sum_{p \in V, q \in N_p} \phi_{pq}(x_p, x_q) + \sum_{c \in S} \phi_c(x_c) \tag{2.14}$$

where, $\phi_p(x_p)$ is the unary potential, $\phi_{pq}(x_p, x_q)$ is the pairwise potential and $\phi_c(x_c)$ is defined over a segment S and is associated with higher order region consistency potential.

The unary potential term is defined as

$$\phi_p(x_p) = \theta_N \phi_N(x_p) + \theta_{band} \phi_{band}(x_p) \tag{2.15}$$

where, $\theta_N \phi_N(x_p)$ denotes the potential due to the output produced by the neural network classifier described in Section 2.3.3 and $\theta_{band} \phi_{band}(x_p)$ is the potential from the band values from the NAIP images.

We can define the potential $\phi_N(x_p)$ derived from the classifier output as

$$\phi_N(x_p) = -\log P(C_p|x) = -\log y_p \tag{2.16}$$

Here, $P(C_p|x)$ denotes the normalized distribution generated by the classifier and y_p denotes the output distribution from the classifier.

Similarly, the pairwise term $\phi_{pq}(x_p, x_q)$ is updated to encode the band information as:

$$\phi_{pq}(x_p, x_q) = \begin{cases} 0, & \text{if } x_p = x_q \\ \theta_P + \theta_V exp(-\theta_\beta ||B_p - B_q||^2), & \text{otherwise.} \end{cases} \tag{2.17}$$

Here, B_p and B_q are the band vectors for pixels p and q respectively. The model parameters θ_N, θ_{band}, θ_P, θ_V and θ_β are derived from the training data using another meta-learning process. The term $\phi_c(x_c)$ denotes the region consistency potential as defined in [58] and is given by:

$$\phi_c(x_c) = \begin{cases} 0, & \text{if } x_p = l_k \\ \theta_R |c|^{\theta_\alpha}, & \text{otherwise.} \end{cases} \tag{2.18}$$

Here, $|c|$ is the number of pixels in the segment and l_k denotes the label assigned to the segment c. $\theta_R |c|^{\theta_\alpha}$ denotes the cost associated with labelings that do not confirm with the labeling of the other pixels in the segment. This term ensures that the labels assigned to the pixels belonging to the same segment are consistent with one another, i.e., it can be said with a high probability that pixels belonging to the same segment belong to the same object category. As illustrated in [58], this is useful for generating object segmentations with fine boundaries and particularly helpful for accurate delineation of tree cover areas in aerial images, where a single pixel denotes an area of $1m^2$. The CRF output with the unary, pairwise and the region consistency terms $\phi_p(x_p)$, $\phi_{pq}(x_p, x_q)$ and $\phi_c(x_c)$ are shown in Figure 2.5. We see that the pairwise term helps improve the classification accuracy of the unary term by reducing the probability values associated with the false positives as evident from the fact that the probability values of most of the yellow (high probability) pixels appearing among the barren patch of land in figure (a) for unary potential are effectively reduced (denoted by blue/purple pixels) by the pairwise term in figure (b). Similarly, the region consistency term (or the segmentation term) improves upon the unary and pairwise terms by reducing the false positives further – most of the blue/purple pixels in figure (b) for the pairwise term are cleaned using the segmentation term/ the region consistency term in figure (c).

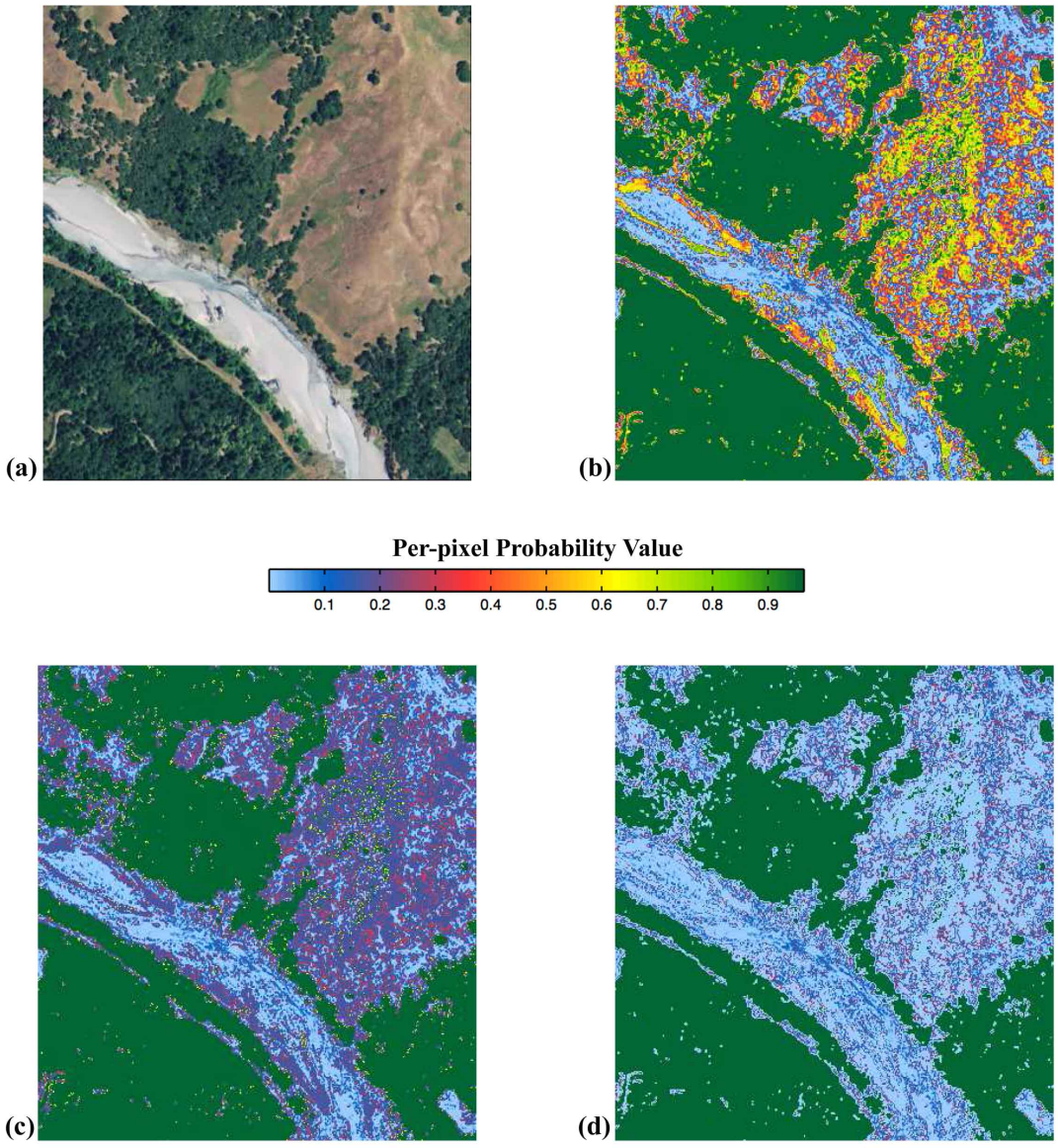


Figure 2.5: (a) A sample NAIP tile and the CRF output probability maps with (b) the unary term $\phi_p(x_p)$, (c) the combination of the unary term $\phi_p(x_p)$ and the pairwise term $\phi_{pq}(x_p, x_q)$ and (d) the combination of the unary term $\phi_p(x_q)$, the pairwise term $\phi_{pq}(x_p, x_q)$ and the region consistency term $\phi_c(x_c)$.

- **The CRF Learning Algorithm**

The energy minimization in CRF is done using the α-expansion algorithm and the $\alpha\beta$-swap algorithm[20]. In the α-expansion algorithm, for a given label α, an arbitrary set of pixels are assigned to this class label. For the $\alpha\beta$-swap algorithm, given a set of pixels with labeling α and another set of pixels with labeling β, the algorithm swaps the class labels for these set of pixels until the energy cannot be minimized any further. Details of the algorithms are provided in Algorithm 1 and Algorithm 2. The key step in both algorithms is Step 5 where $\hat{y}$ is computed using graph cuts[20].

Algorithm 1 Alpha Expansion Algorithm

1: **procedure** ALPHAEXPANSION
2: Assign an arbitrary labeling y to the pixels of the image.
3: *done* $\leftarrow 0$
4: **for** `each input label` $\alpha \in L$ **do**
5: calculate $\hat{y} \leftarrow$ argminE(y') out of y' where y' lies in the span of one α-expansion of y
6: **if** $E(\hat{y}) < E(y)$ **then**
7: y $\leftarrow \hat{y}$
8: *done* $\leftarrow 1$
9: **if** *done* = 1 **then**
10: **goto 3**.
11: **return** y.

Algorithm 2 Alpha-Beta Swap Algorithm

1: **procedure** ALPHABETASWAP
2: Assign an arbitrary labeling y to the pixels of the image.
3: *done* $\leftarrow 0$
4: **for** `each set of label pairs` $\alpha, \beta \in L$ **do**
5: find $\hat{y} \leftarrow$ argminE(y') among y' where y' lies within one $\alpha\beta$-swap of y
6: **if** $E(\hat{y}) < E(y)$ **then**
7: y $\leftarrow \hat{y}$
8: *done* $\leftarrow 1$
9: **if** *done* = 1 **then**
10: **goto 3**.
11: **return** y.

- **Learning the Model Parameters**

The optimal values of the model parameters were learnt by cross validation error minimization of the final pixel labeling assigned to the validation image set. Multiple rounds of cross-

validation were used where different subsets of the images were chosen for training and validation. The combined space of the parameter values θ_N, θ_{band}, θ_P, θ_V, θ_β and θ_α of the CRF is exponential in the size of the parameter space and it is computationally intractable to determine the optimal values by performing an exhaustive search over the space of parameter values. A heuristic approximation technique was used by first optimizing the unary model parameters θ_N and θ_{band} followed by the parameters θ_P, θ_V and θ_β for the pairwise potential terms and finally the higher order parameters θ_R and θ_α.

2.3.5 Online Update of the Training Database

Once the final results are obtained, the training database is updated online with incorrectly labeled examples using expert knowledge on the fly. It should be noted that "expert knowledge" here means using humans with domain knowledge to hand-label image patches related to various landcover classes. This is done as follows – After the generation of tree-cover maps from a certain number NAIP tiles (100, here), 10 (10% in general) maps are chosen at random and a reference to the NAIP tiles corresponding to these maps are saved to a database. An automated image-rendering tool (developed as part of our framework) allows experts to re-label misclassified image patches. The details of this image relabeling tool are provided in Section 2.3.6. These re-labeled patches are then saved to the training database with the correct labeling. This improves the quality of results produced by the classifier in subsequent iterations. Choosing 10% of the image tiles randomly after the generation of every 100 tiles helps in maintaining the homogeneity of candidate selection for relabeling among the generated probability maps. 100 consecutively processed NAIP tiles cover a relatively small geographical area. Hence, a random 10% of the tiles represent a uniform selection of tiles from every spatial window and choosing every 100 images ensures a uniform selection from the entire mapped region. Every time the training dataset is updated, automated online training is done. This online update stage is very similar to a supervised post-processing of the classifier just like the boosting algorithm in machine learning [87] which recursively updates a strong learner by higher re-weighting of misclassifications by weak learners. The online update phase helps in reducing the False Positive Rate while significantly

increasing the True Positive Rate. Figure 2.6 lists the variation of the User's and Producer's accuracy (commission errors and omission errors) with changing epochs of the online update algorithm. It can be seen that both the User's and Producer's accuracy improve with the number of re-training epochs for the online update algorithm until 8 to 10 iterations. After that both accuracies remain stable up to about 14 epochs, after which they start dropping. This can be attributed to overfitting of the classification algorithm due to an excessive number of training samples fed into the supervised learner.

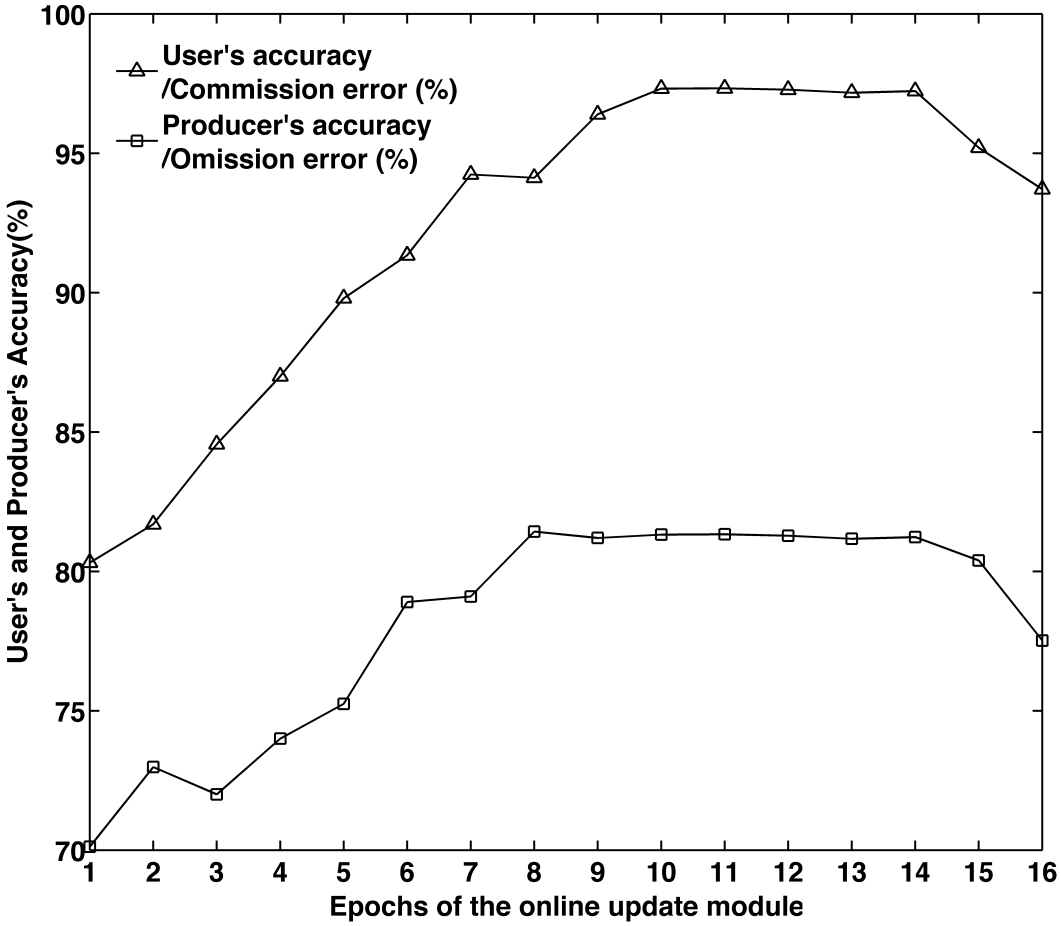


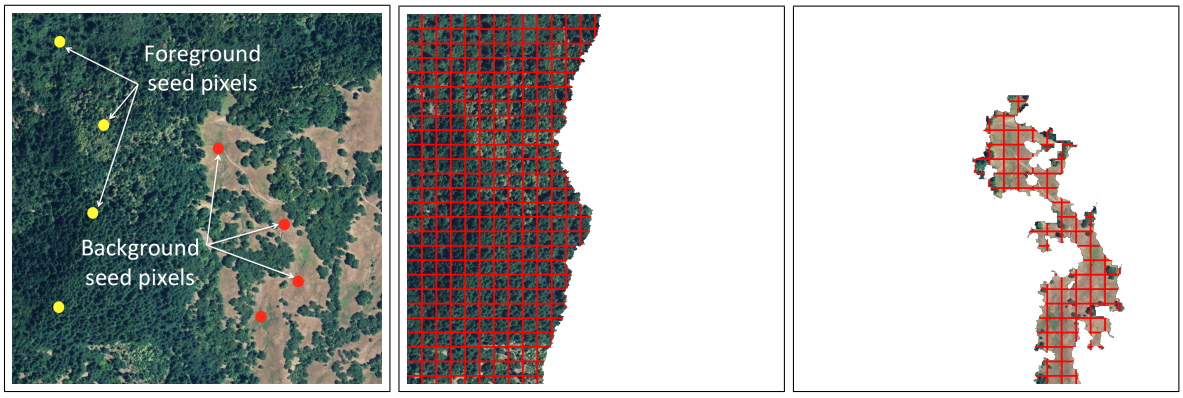
Figure 2.6: Variation of Omission and Commision Errors with changing epochs of the online update algorithm.

2.3.6 Image Labeling and Re-labeling using Interactive Segmentation

We use an interactive image segmentation tool to extract and label the training samples as well as to re-label misclassified image patches in the online update phase described above. The interactive segmentation module uses a Random Walk based image segmentation algorithm first presented in [42]. In this method, at first a certain number of pixels are labeled as background and foreground seed pixels. These act as seeds for the segmentation algorithm. For any given unlabeled pixel in the original image, a random walk is initialized at the pixel. It is possible to estimate the probability with which a random walk which starts at any unlabeled pixel will reach one of the forground or background seed pixels first. For k seed pixels, we get a $k \times 1$ probability vector for each unlabeled pixel, each element of which represents the probability that the random walk starting at that particular pixel reaches the corresponding seed pixel first. Then we can label each pixel as belonging to a specific class based on which element in the probability vector has the highest value. Figure 2.7 shows a sample NAIP tile with tree and non-tree masks generated by the Random walk based segmentation algorithm by selecting a certain number of foreground and background seed pixels corresponding to tree and non-tree areas. The foreground and background seed pixels are marked with yellow and red circles, respectively. The red boxes in Figure 2.7(b) and Figure 2.7(c) represent the training samples extracted from the image which are in turn saved to the training database with the correct label. Note that only complete boxes representing 4×4 training images are saved to the database while the rest are discarded. It is useful to note that the class masks shown in Figure 2.7 are created by manually selecting a certain set of seed pixels. Choosing a different set of seeds can create a different segmentation mask.

2.3.7 Implementation details and the High Performance Computing Architecture

We have deployed the abovementioned modules as stand alone on the NASA Earth Exchange (NEX) supercomputing cluster. The deployment was done through QSub routines and the Message Passing Interface (MPI). The data was accessed through a MySQL database. The NAIP tiles were processed in parallel in the cores of the NASA Earth Exchange High Performance Com-



(a) Sample NAIP tile (b) Tree mask (c) Non-tree mask

Figure 2.7: A sample NAIP tile with tree and non-tree cover masks generated by the Random Walker Segmentation module by selecting a certain set of foreground and background seed pixels. The small red squares indicate the training samples extracted from the image which are in turn saved to the training database with the correct label. Note that only complete squares representing 4×4 training images are saved to the database while the rest are discarded.

puting (NEX HPC) platform. Each node in the cluster having Harpertown CPUs consists of 8 gigabytes of memory and 8 cores with 3GHz processors per node [111]. In order to process 8 tiles in parallel, one tile per core, the memory requirement per core has to be kept lower than 1 Gigabyte. However, the problem arises with the use of the Statistical Region Merging (SRM) algorithm illustrated in Section 2.3.1. Despite being fast, the algorithm has to store all the indices in memory while sorting them using radix sort as it makes decisions about region boundaries using global scene level image descriptors. This has space complexity of the order of $O(n^2)$, which indicates all image gradients in a $n\times n$ image, which is of the order of $\sim$3 Gigabytes for a typical NAIP tile. In order to address this memory-performance tradeoff, each image was split into $\lambda\times\lambda$ windows and then fed in a pipeline to each core in the HPC node. λ was chosen to be 256 for our experiments, because, higher values led to a higher memory requirement while lower values resulted in a substantial increase of processing time. The current architecture takes a maximum of approximately 4 hours to process each NAIP tile. The details of the architecture are illustrated in Figure 2.8.

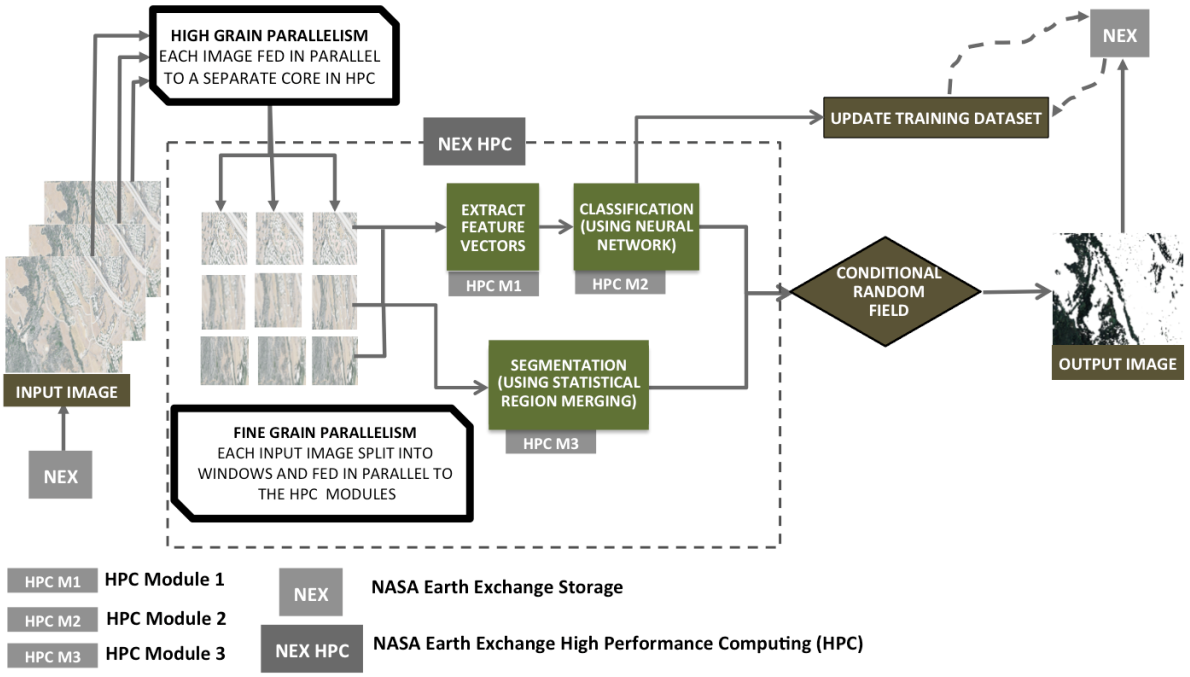


Figure 2.8: High Performance Computing Architecture of our approach.

2.4 Results and Discussion

A rudimentary implementation of our framework produced encouraging results. We chose 1500 image tiles at random covering the following three types of landscapes (1) Densely forested, (2) Fragmented forests, and (3) Urban forested areas. A total of 36000 sampling points were chosen at random from the test images - 12000 samples for each land-cover type. The sample validation points are shown in Figure 2.19. The classifier accuracy was measured and averaged over 100 iterations. The results are tabulated in Table 2.4, which shows that our framework produces true positive rates higher than 85% for both densely forested and fragmented areas. However, the results degraded for urban areas where we achieved correct detection rates of about 74%. This can be attributed primarily to the presence of trees in urban regions with canopies having dimensions of less than 4m in any direction (the value of our neighborhood parameter τ). However, experimenting with τ values less than 4 did not improve the performance of the framework as evident from the ROC Curve analysis presented in Figure 2.9. The ROC curves represent

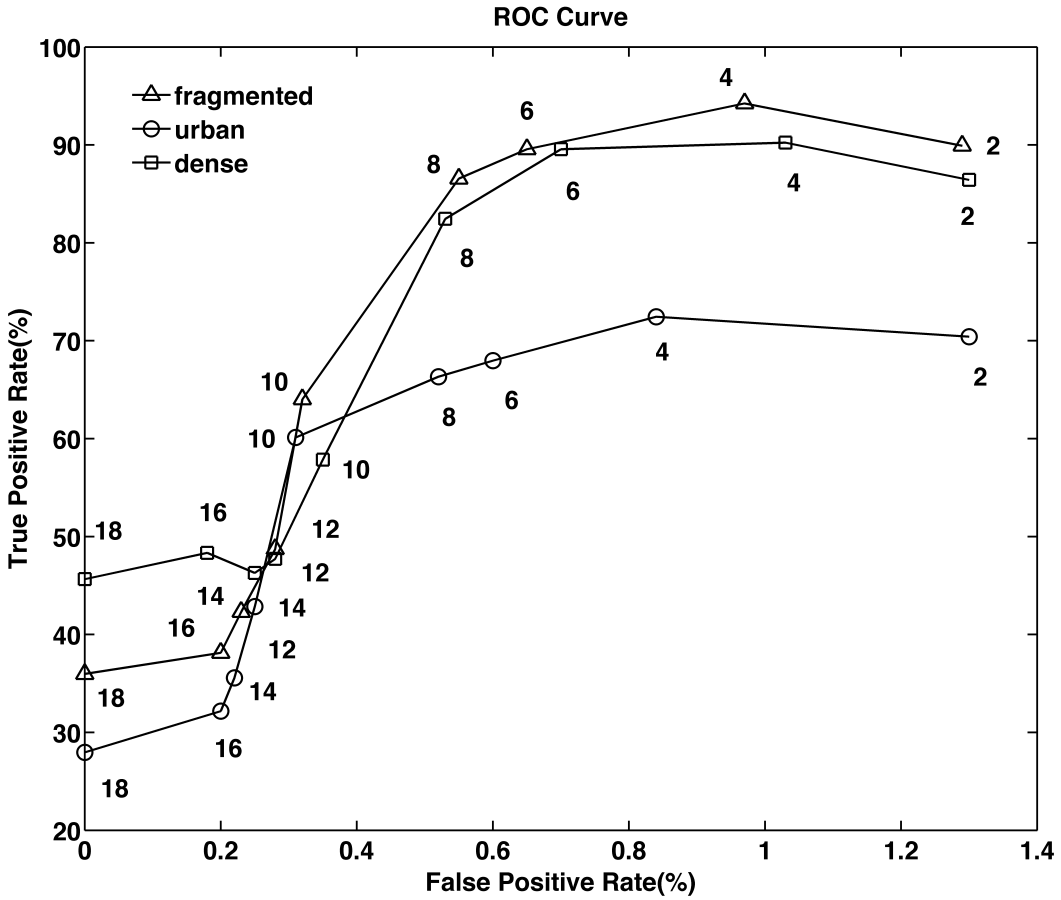


Figure 2.9: ROC Curves for the three types of landscapes considered – fragmented, urban and densely forested (The numbers indicate the corresponding window sizes). One representative NAIP tile was chosen for each landscape – a densely forested tile from the Gasquet region in Northwestern California (7750m $\times$ 6380m), a tile with fragmented forests from the Susanville region in Northeastern California (7610m $\times$ 6000m), and an urban tile from a region in San Jose, California (7620m$\times$6240m). 5000 points were chosen randomly from each image tile, labels for these representative points were assigned by a human expert and the tree cover maps were validated against these ground truth data to generate the true positive rate and false positive rate for the ROC Curves.

the change in True Positive Rate with the change in False Positive Rate while varying a certain adjustable parameter of a model (window size τ here). The degradation in performance of the framework after the adjustable parameter τ is increased beyond a certain value (4, here) can be attributed to a flat response from the classification module due to reduced discriminative power of the class separability criterion. Table 2.5 shows the confusion matrix, where, the columns

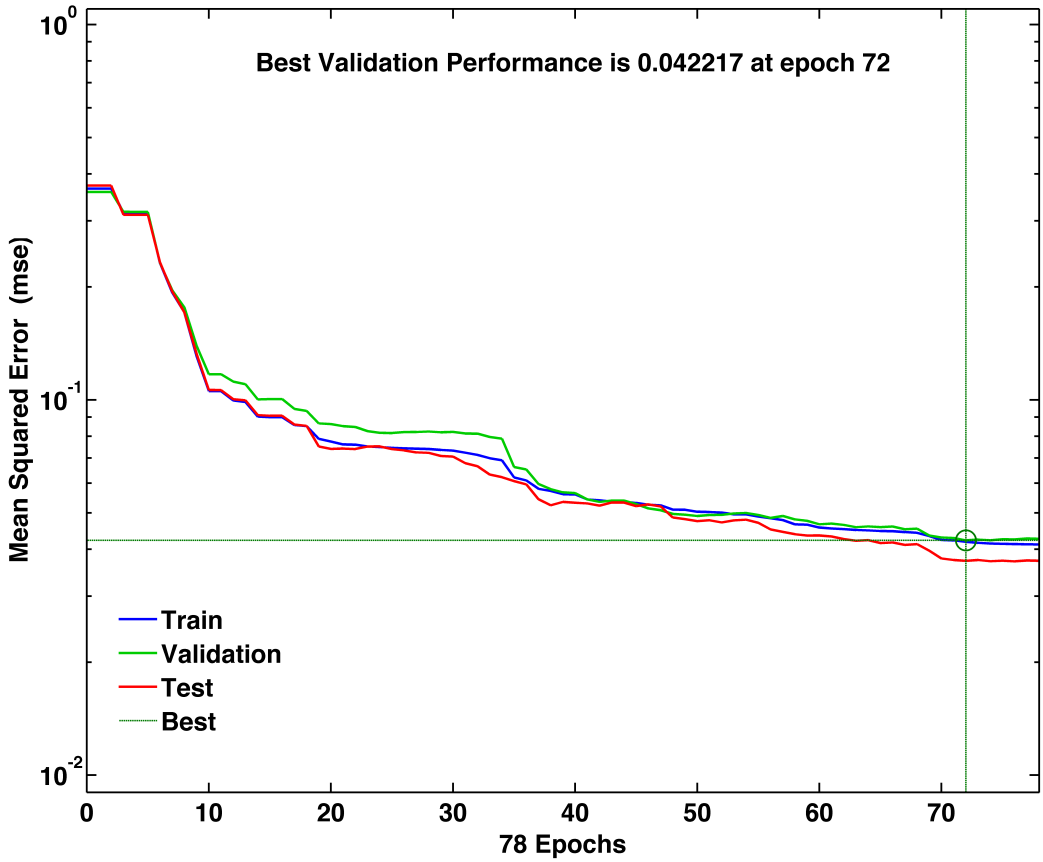
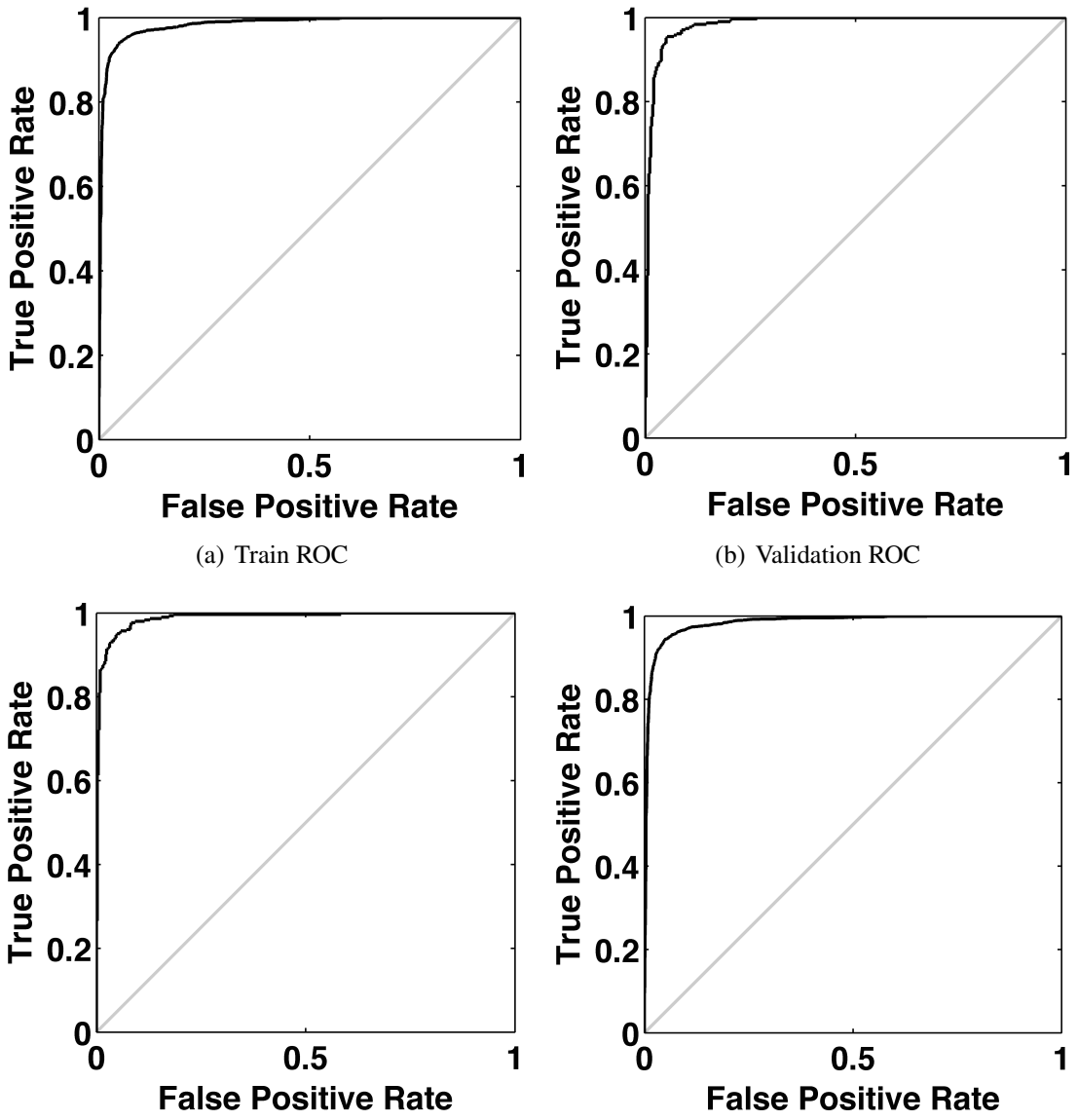


Figure 2.10: Performance of the Neural Network training algorithm for a set of (randomly chosen) 3500 training samples, 750 validation samples and 750 test samples from a NAIP tile from Blocksburg, California (7610m $\times$ 6000m). The X-axis marks the iterations/epochs of the training algorithm, while the mean-squared error is noted along the Y-axis. The blue line indicates the mean squared error at various epochs during the training phase of the Neural Network, the green line indicates the mean squared validation error and the red line indicates the mean squared test error. The best performance is attained at iteration 72.

represent the instances of the actual class while the rows represent the instances of the predicted class. It can be seen from the table that our framework produces an overall accuracy of 90.31%. ROC Curves generated by changing the neighborhood parameter τ (defined in Section 2.3.3) for various land-cover types (fragmented forests, densely forested and urban areas) is shown in Figure 2.9. τ was chosen to be 4 by doing a ROC Curve analysis as illustrated in Figure 2.9, such that the True Positive Rate is maximized. The error minimization of the Neural Network training



(a) Train ROC

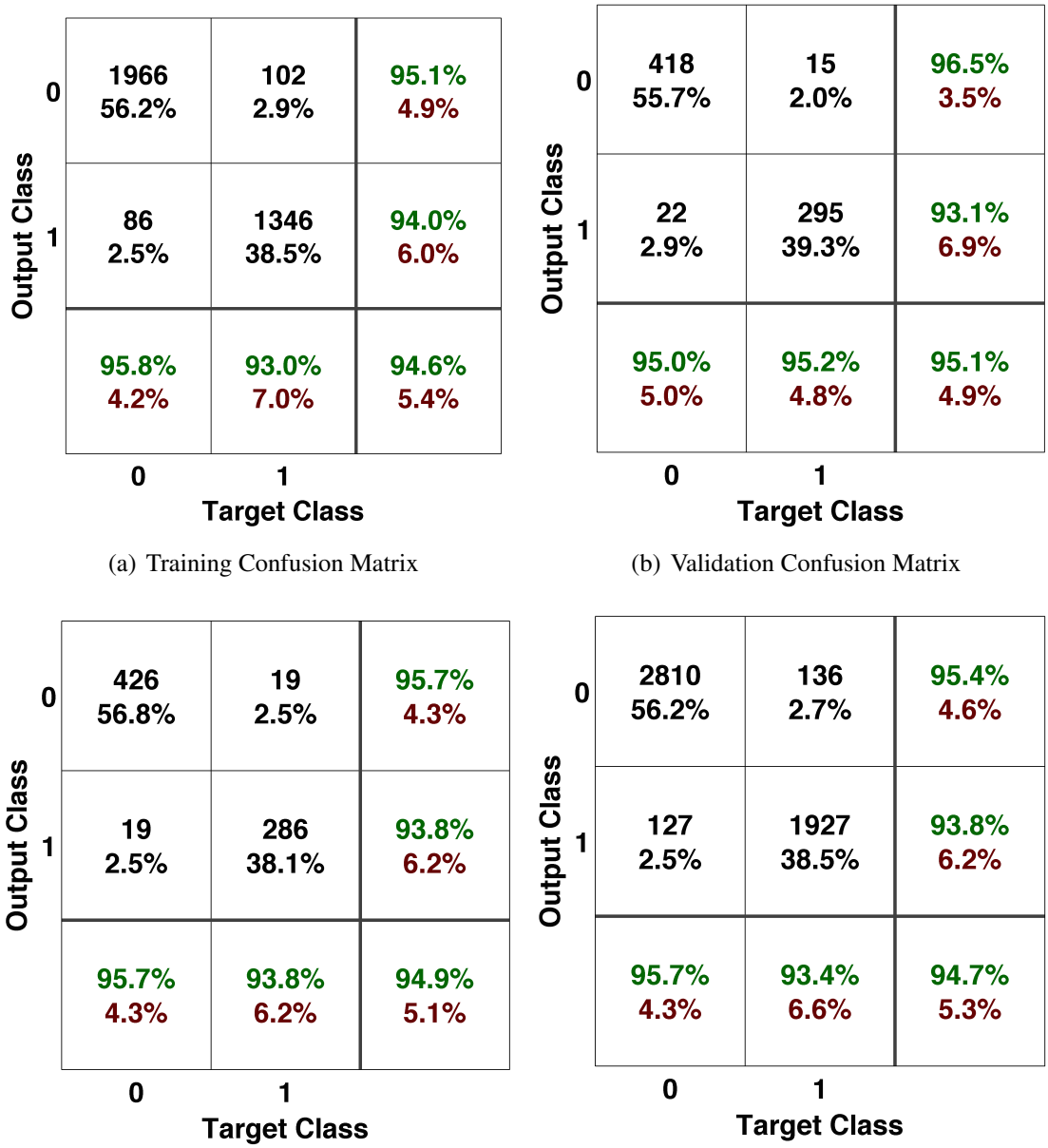
(b) Validation ROC

(c) Test ROC

(d) All ROC

Figure 2.11: ROC Curves for the Neural Network Training algorithm for the same dataset used in Figure 2.10 – ROC curve generated for training, validation, test sets and taking the mean of the true positive rate and false positive rate of the training, validation and test dataset respectively.

algorithm is shown in Figure 2.10. It can be seen that the best validation performance with a mean-squared error of 0.14191 is attained at epoch 30 of the Neural Network training algorithm. Figure 2.11 and Figure 2.12 show the ROC Curves and confusion matrix for the Neural Network training algorithm. Comparative studies with the 2001 National Land Cover Data (NLCD) [99] 30-m are enumerated in Table 2.6. It can be seen that results from our probabilistic framework



(a) Training Confusion Matrix

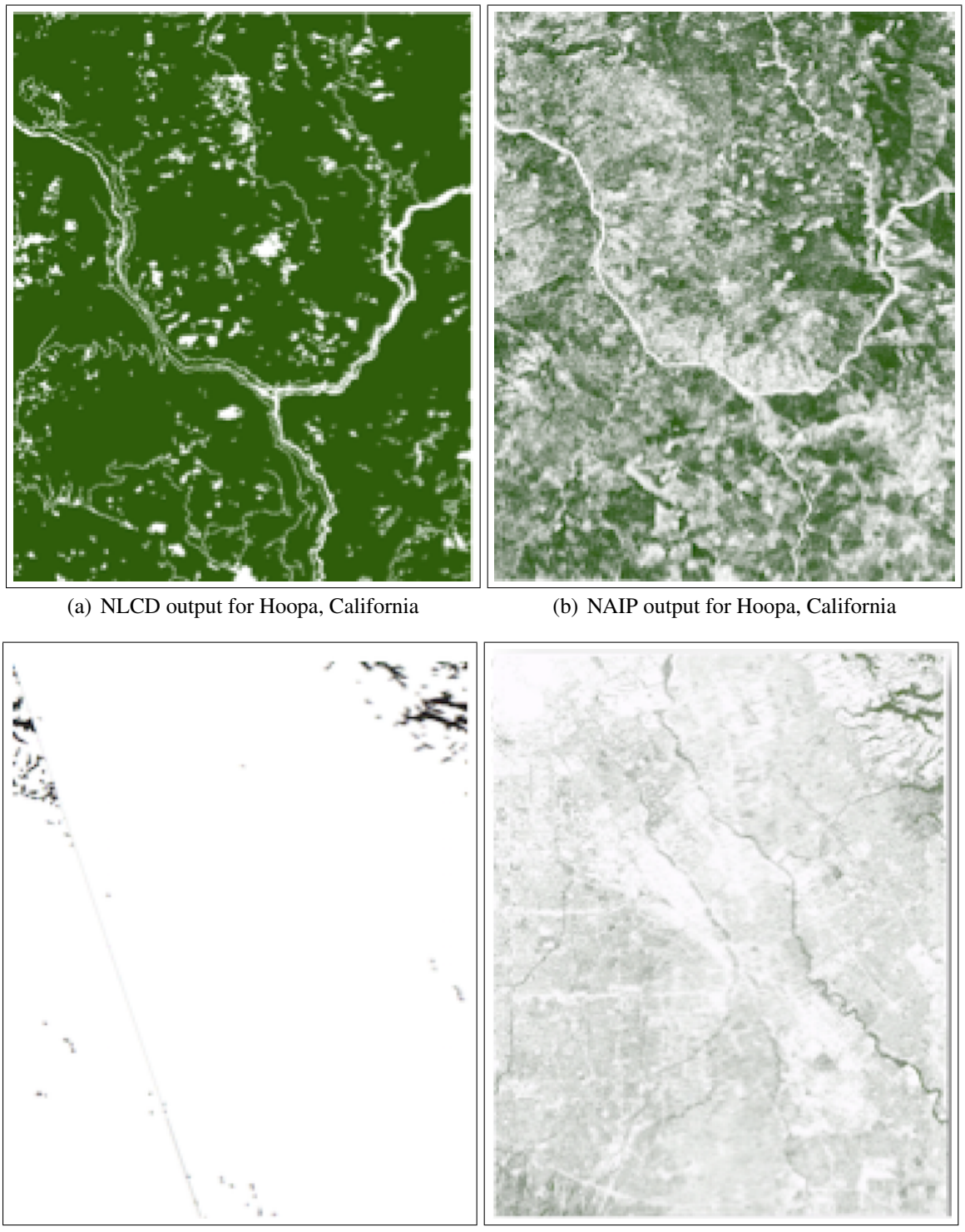
(b) Validation Confusion Matrix

(c) Test Confusion Matrix

(d) All Confusion Matrix

Figure 2.12: Confusion Matrix for the Neural Network training algorithm for the dataset used in Figure 2.10 and Figure 2.11. A total of 3500 samples are used for training, 750 samples for validation and 750 samples for testing.

outperforms NLCD by nearly 15% for fragmented forests and nearly 77% for urban areas. Figure 2.13 shows the sample output for two tiles - one for a fragmented area in Hoopa, California, north of the Klamath River and another for an urban area in San Jose, California using NLCD and NAIP. The comparative studies with NLCD clearly show that our algorithm outperforms the



(a) NLCD output for Hoopa, California

(b) NAIP output for Hoopa, California

(c) NLCD output for San Jose, California

(d) NAIP output for San Jose, California

Figure 2.13: Results for an image tile with fragmented trees in Hoopa, California, north of the Klamath River and an urban area in San Jose, California for NLCD and NAIP.

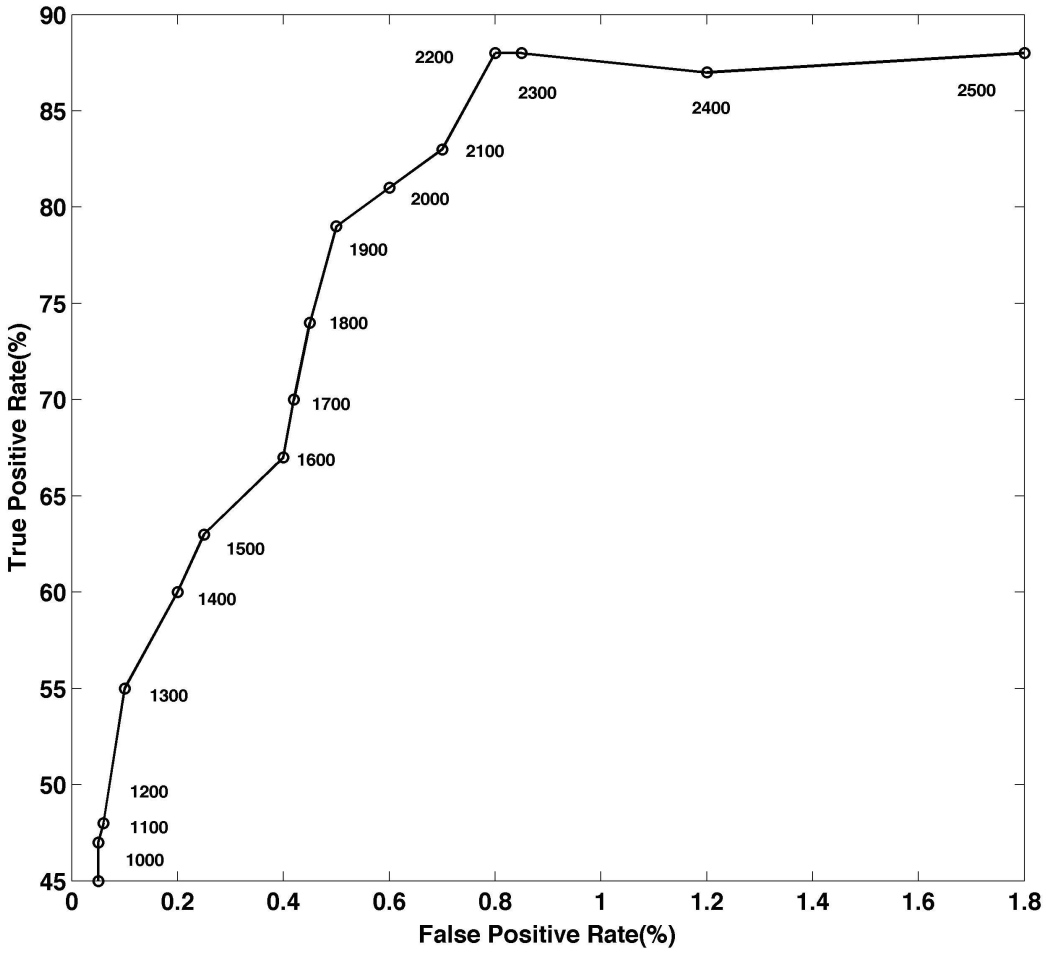


Figure 2.14: ROC Curve generated by changing the sample size of the training data. The region of study was the same as that of Figure 2.10 – an area with fragmented forests in the Blocksburg region in northwestern California (7610 m $\times$ 6000 m). The numbers denote the number of training samples.

NLCD approach for the classes of land-cover types (tree-cover areas) considered in this study. This can be primarily attributed to the higher resolution of the NAIP dataset as compared to LANDSAT imagery which has 30 times lower resolution than NAIP. Moreover, our structured prediction framework helps us in decreasing false positives by considering intra and inter-class votes towards the labeling algorithm. Figure 2.14 shows the ROC Curve generated by changing the sample size of the training dataset from 1000 samples per class to 2500 per class in steps

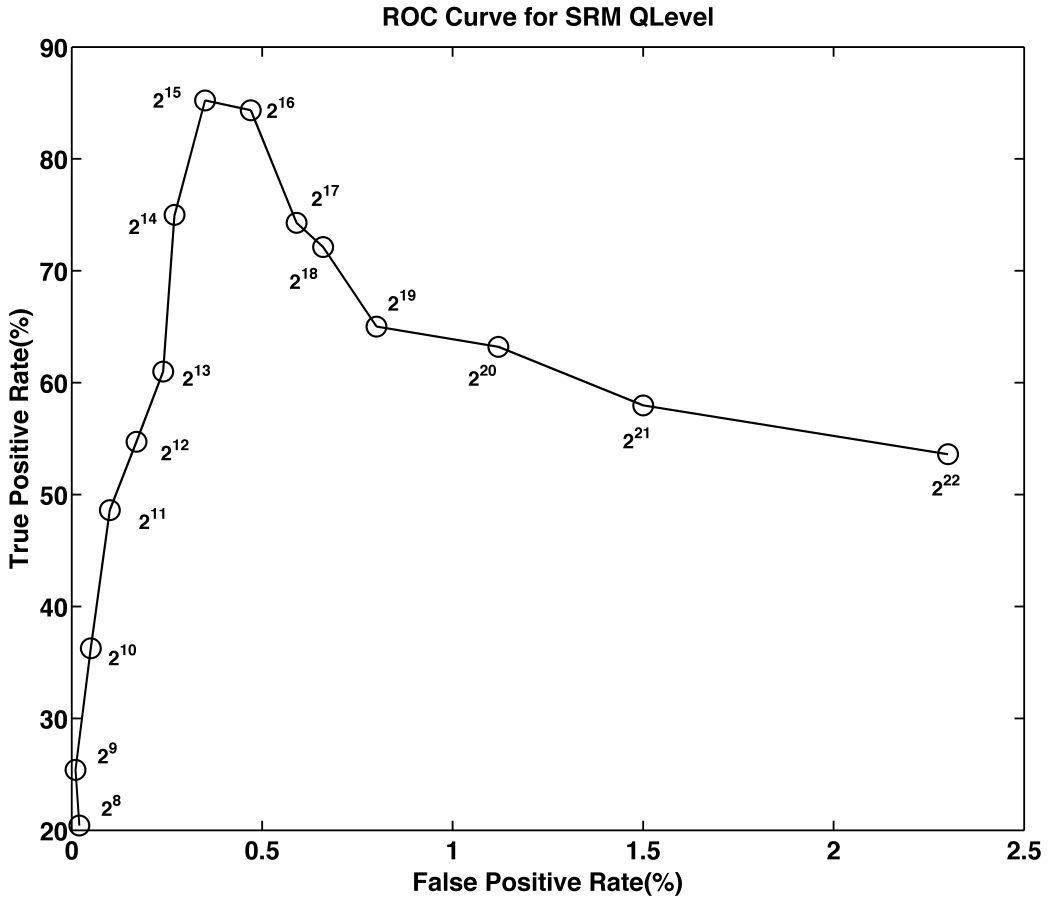


Figure 2.15: ROC Curve generated by changing the Quantization Level in the SRM algorithm for the same area as considered in Figure 2.14. The training sample size is 5000 - consisting of 2500 tree and 2500 non-tree samples chosen at random from the NAIP tile. The numbers denote the corresponding Qlevel values.

of 100 for a particular tile in the NAIP dataset with fragmented forest cover. The flat response towards the end of the curve indicates that increasing the training sample size has minimal effect beyond a point, which is around 2200 training samples for this exercise. This shows the robustness of the algorithm and the fact that minimal amount of training samples is sufficient for training the classifier. Figure 2.15 shows a ROC Curve generated by changing the Quantization Level (Qlevel) in the Statistical Region Merging algorithm. The analysis of this curve helped us in selecting a Qlevel of 2^{15} for our framework as is evident from the maxima attained at $Q = 2^{15}$ in the figure. Figure 2.16 shows the probability maps generated by the classification algorithm

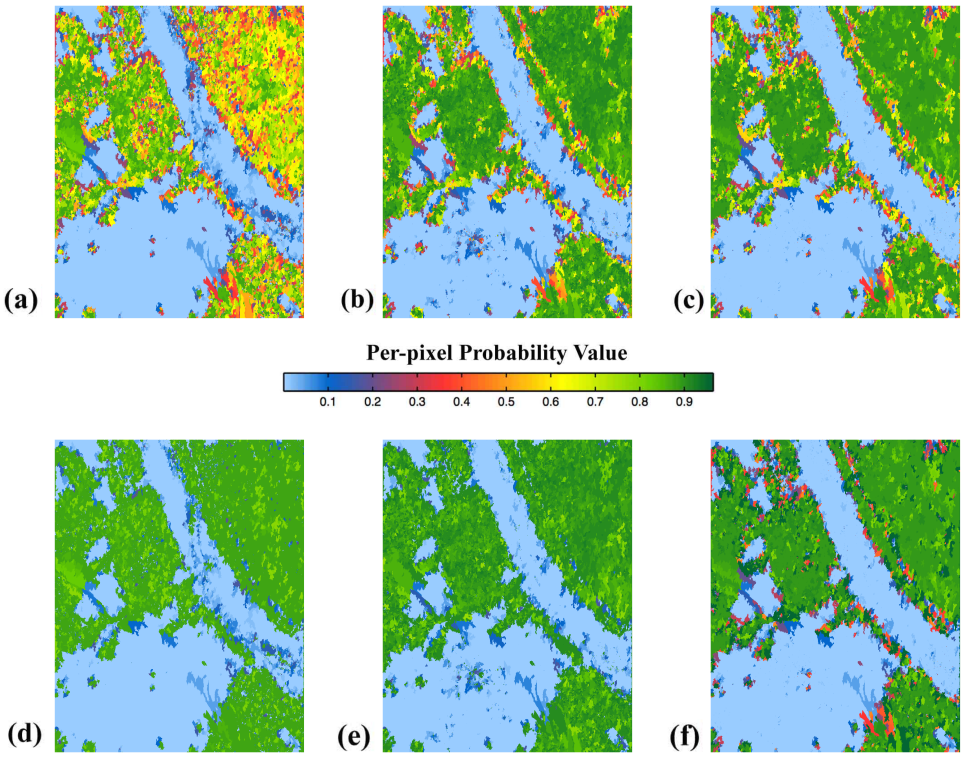


Figure 2.16: Probability Maps for the probabilistic NN classifier results (a-c) and CRF output (d-f) for various training sample sizes (1300,1800 and 2500 samples per class from left to right) for a sample NAIP tile from Blocksburg, California. The color maps show the probabilities on a scale of 0 to 1. The probability maps for the NN represent the probability of a pixel being predicted as a tree by the Neural Network and the probability maps for the CRF output represent the final probabilities assigned to the pixels by the CRF labeling algorithm. A pixel assuming a value of 1 in the probability map is marked as a tree and a pixel assuming a value of 0 is marked as a non-tree, with intermediate pixels values being marked as tree/non-tree according to the problem (here, we use a 50% threshold, i.e., a pixel is marked as tree if the probability exceeds 0.5).

and the Conditional Random Field. Figure 2.17 shows the final probability map generated for a sample tile from the NAIP database using our framework. Figure 2.27 shows a sample NAIP tile from the Blocksburg area in California and the corresponding binary tree-cover map generated by our framework. To generate this binary tree cover map, the output probability map from the CRF is filtered with a threshold τ to eliminate pixels with output probability less than the threshold. The threshold τ is set to 0.5. For the CRF output probability for a pixel x being $Pr(x)$, the final

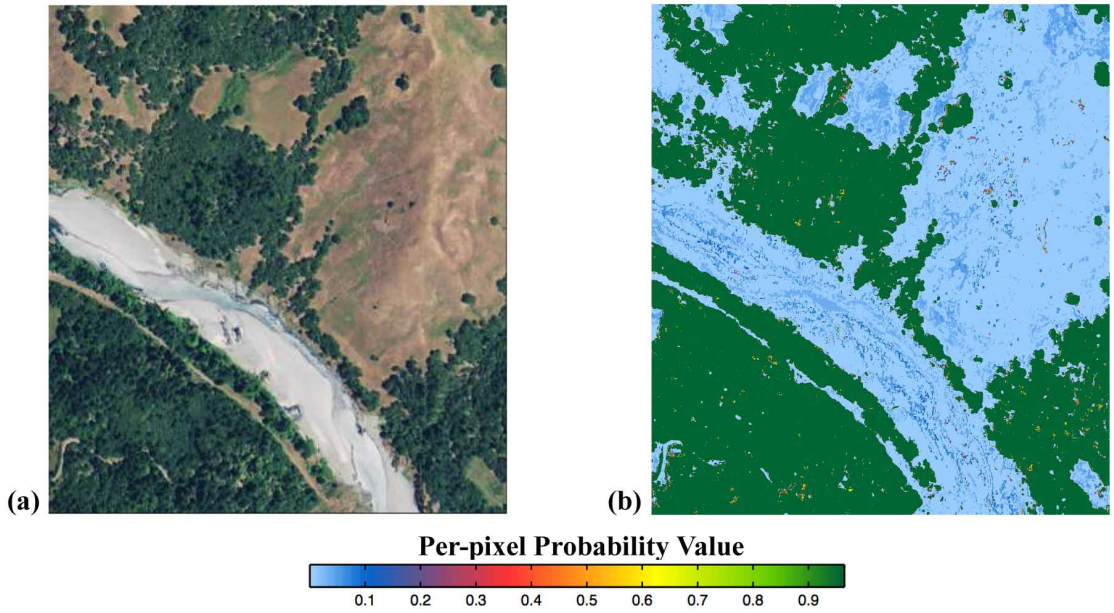


Figure 2.17: (a) A sample image with (b) the final Probability Map generated by our framework for a region in Blocksburg, California. This final probability map is the same as the map generated by the CRF based labeling algorithm as shown in Figure 2.16. The CRF algorithm combines the probability values assumed by the classifier outputs for individual pixels and generates the final probability map as shown above.

output map value for pixel x,

$$O(x) = \begin{cases} 1, & \text{if } Pr(x) \geq 0.5 \\ 0, & \text{otherwise.} \end{cases} \tag{2.19}$$

Figure 2.28 shows the final tree-cover map generated by our framework for the whole of California covering 11,095 NAIP tiles.

2.4.1 Validation with High Resolution Airborne LiDAR Canopy Height Model

The tree-cover maps generated by the Canopy Height Model (CHM) from the LiDAR data and the probabilistic framework for the NAIP dataset for both Area 1 and Area 2 are presented in Figure 2.20.

A Random Forest (RF) based classifier was independently trained on the same dataset used

Network Arch. Neurons/layer [Layers]	Classifier Accuracy 150 features(%)	Classifier Accuracy 22 features(%)
10 [2]	88.81	90.59
20 [2]	88.9	91.63
50 [2]	90.16	92.24
100 [2]	89.34	89.93
10 [3]	84.53	87.97
20 [3]	83.72	85.23
50 [3]	89.4	86.42
100 [3]	88.85	76.015

Table 2.3: Classification Accuracy of the classifier with various network architectures using the entire set of 150 features and the set of 22 features derived using the feature selection method presented in Section 2.3.2

	Densely Forested	Fragmented Forests	Urban Forests	Overall
Total Samples	12000	12000	12000	36000
Tree Samples	6000	6000	6000	18000
Non-tree Samples	6000	6000	6000	18000
True Positive Rate	85.87	88.26	73.65	82.59
False Positive Rate	2.21	0.99	1.98	1.73

Table 2.4: Preliminary classification accuracy assessment.

for the probabilistic Neural Network classifier. The RF classifier was implemented using a random forest package [73], available in the R interface [112]. The number of trees, node size and maximum number of terminal node trees in the random forest were varied in iterations to achieve a stable solution with maximum accuracy. The number of trees was set to 250. Each node was set to size of 5 and maximum number of terminal nodes was set to 500. Number of trees is selected in such a way so that every input row gets predicted a few number of times at the least. The cardinality of the set of terminal nodes decides the maximum possible size of the growth of the trees (also subject to limits by node size). The final values of the parameters were obtained empirically depending on the minimum execution time and memory requirements when further parameter variations did not increase the accuracy. A sliding window analysis (with a window size of 50 pixels) was performed on the two scenes and the percentage of tree-cover pixels and non-tree pixels are presented in Figure 2.21 and Figure 2.22. As seen from Figure 2.21, the

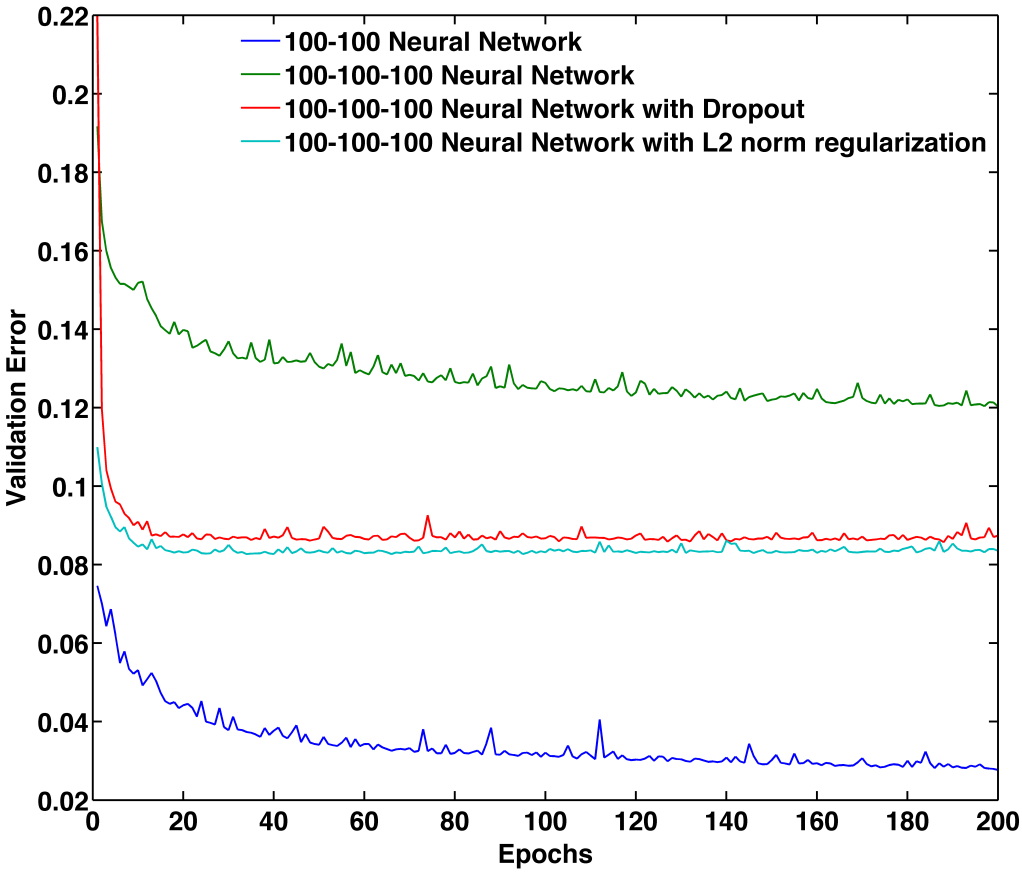


Figure 2.18: The validation error rate for the same dataset used in Figure 2.10 for the 100-100 and 100-100-100 neural networks without regularization, and the 100-100-100 neural network with L2 norm regularization and Dropout.

tree-cover predictions generated by both the Probabilistic Neural Network (NN) framework and the Random Forest (RF) based framework have a high positive correlation with LiDAR, while NLCD produces significantly less accurate results, having error rate more than 40% on average with LiDAR tree cover estimates considered as ground truth, while the Neural Network classifier produces a mean error rate less than 5% with the same ground truth data. The Random Forest implementation has a higher error rate averaging around 15%. An evaluation of the True Positive and False Positive Rates for the NN and RF algorithms for the NAIP data and NLCD algorithm for LANDSAT data for Area 1 is enumerated in Figure 2.23(a) and 2.23(b). The significantly higher values of the True Positive Rates for NLCD can be attributed to the significant loss in the

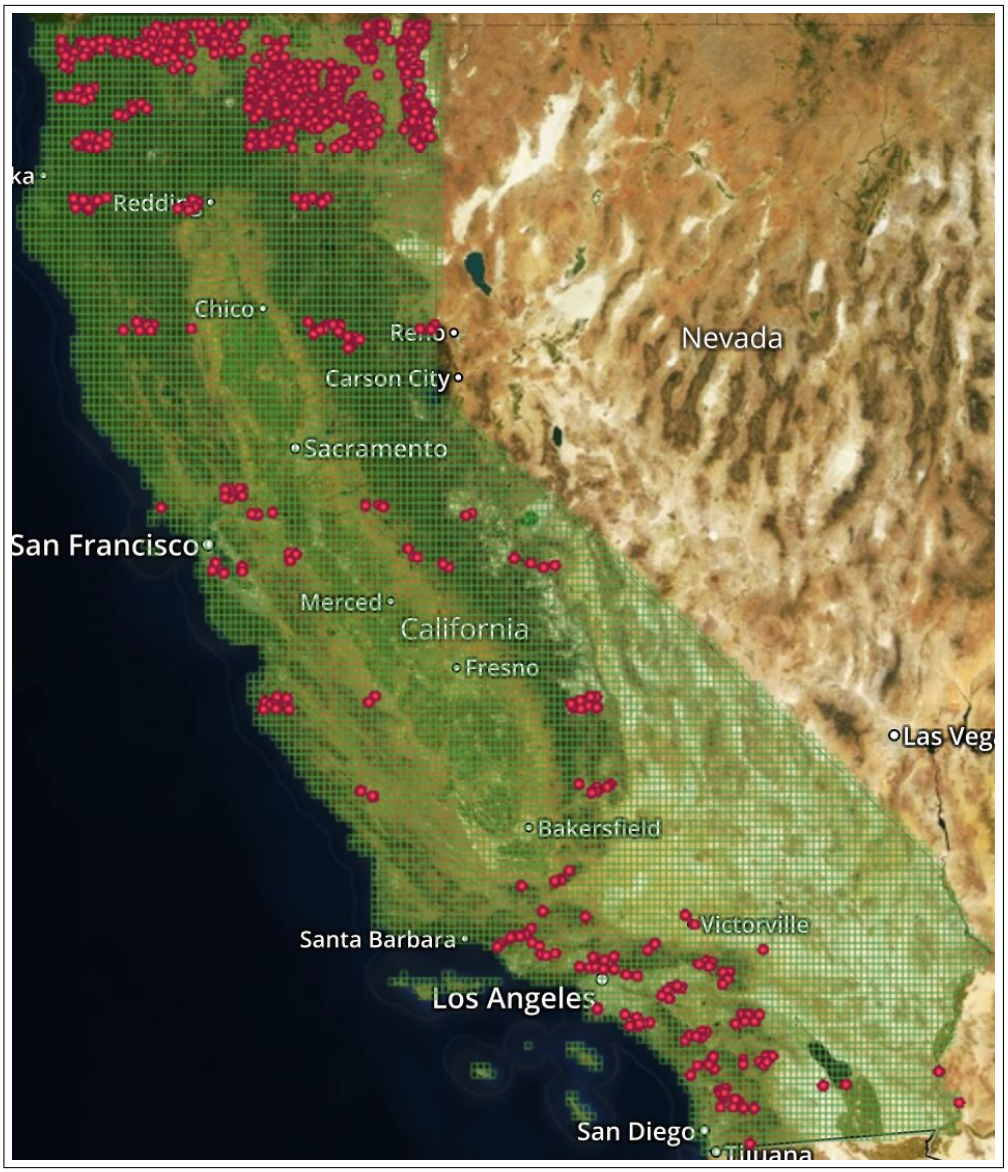


Figure 2.19: A satellite image showing the validation points chosen for our experiments over California. The red circles denote the validation points. A total of 36000 sampling points were chosen to represent densely forested areas, fragmented forests and urban forested areas of California. The green grid represents the individual NAIP tiles. In order to display the locations from which the validation points were sampled, multiple points were clustered into subgroups and hence each red circle in the figure represents multiple validation points.

	Actual Class			
Predicted class	**Tree**	**Non-tree**	**Total pixels**	**User's accuracy**
Tree	14832	317	15149	97.9%
Non-tree	3168	17683	20851	84.8%
Total pixels	18000	18000	36000	
Producer's accuracy	82.4%	98.23%		90.31%

Table 2.5: Confusion Matrix

	NLCD-30m	**NAIP-1m**
Total samples	1000	1000
Tree samples	500	500
Non-tree samples	500	500
True Positive Rate(%)	72.31	87.13
False Positive Rate(%)	50.8	1.9

	NLCD-30m	**NAIP-1m**
Total samples	1000	1000
Tree samples	500	500
Non-tree samples	500	500
True Positive Rate(%)	2.88	79.64
False Positive Rate(%)	3.23	1.68

Table 2.6: Comparative results with NLCD for fragmented forests (top) and urban forested areas (bottom).[113]

resolution of the dataset. Therefore, a tree-cover region is classified in its entirety and approximated as either a tree-cover or non-tree region based on whether most of the pixels are tree or not. In other words, in NLCD, a single pixel represents an area of 900 sq. m. The highest resolution attainable for NLCD is of the order of at least 8-10 full-grown trees. This argument can be substantiated by the NLCD output shown in Figure 2.13. As can be seen in Figure 2.13, non-tree regions are approximated as tree-cover regions in a comparatively densely forested region while for urban areas, tree-cover regions are classified as non-tree regions owing to its presence amidst a sparsely forested landscape. Figure 2.13 explains the high False Positive Rate for NLCD as illustrated in Figure 2.23(b). The same validation process was used for Area 2 and the results of the percentage of tree and non-tree pixels are presented in Figure 2.24 and Figure 2.25. As highlighted by both figures, our probabilistic Neural Network based framework produces near optimal results, which are highly correlated to the LiDAR output and outperforms the Random

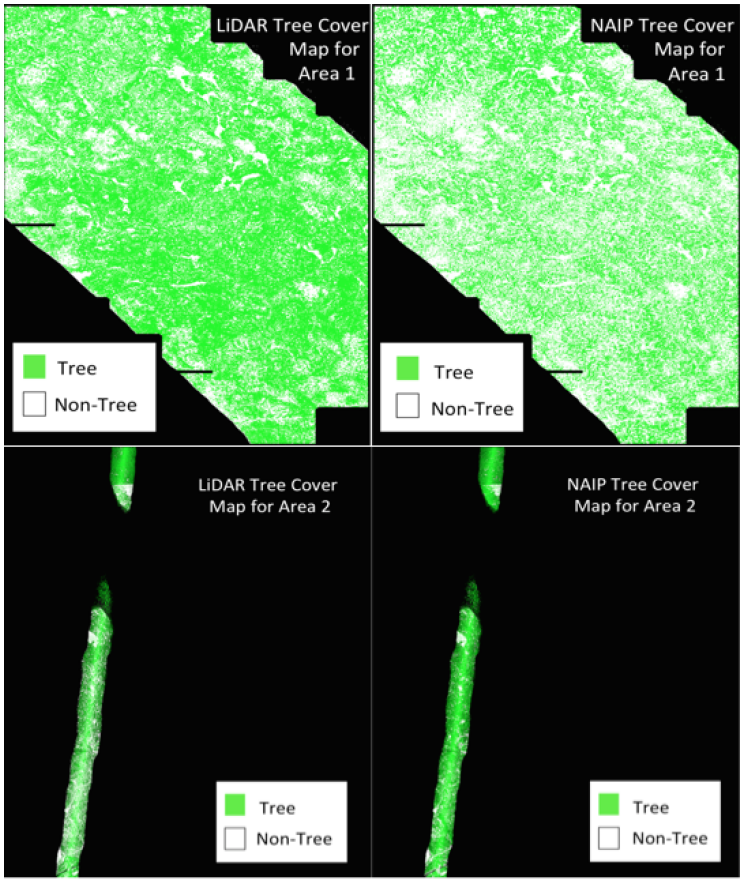


Figure 2.20: The final tree-cover maps generated using LiDAR (left) and NAIP (right) for Area 1 (top) and Area 2 (bottom). The green regions represent tree cover areas, the white regions represent non-tree areas and the black regions represent the areas with null values in the LiDAR data (these black regions were masked out from the NAIP tree cover maps for comparative studies with the corresponding LiDAR maps).

Forest based classification algorithm. This is substantiated by the True Positive and False Positive Rates enumerated in Figure 2.26(a) and 2.26(b). It can be easily seen that the algorithm produces a significantly high accuracy with mean True Positive Rate as high as 97% and mean False Positive Rate around 8%. Though NLCD produces True Positive Rate of around 92% (almost same as NN) for the area, but it produces False Positive Rate as high as 88%. It also outperforms the Random Forest classifier, which produces True Positive Rate of around 82%.

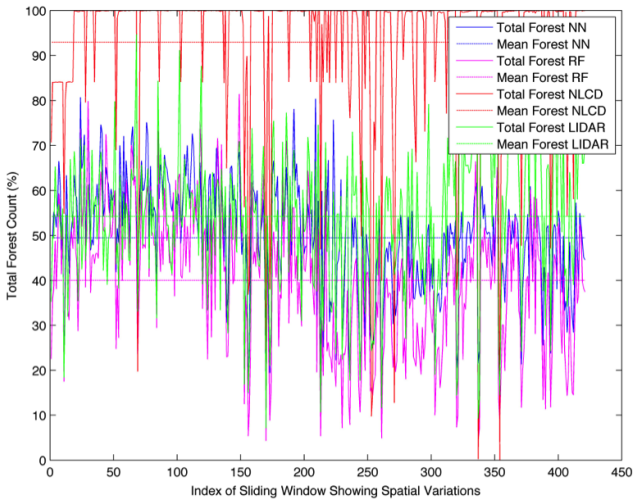


Figure 2.21: Percentage of forest cover obtained using Neural Network and Random Forest (for NAIP), and NLCD and LiDAR in Area 1 (the western Sierra Nevada mountain range over the Teakettle Experimental Forest in California). A 50×50 sliding window was used to obtain the percentage of tree-cover pixels in both NAIP and NLCD with LiDAR as the ground truth.

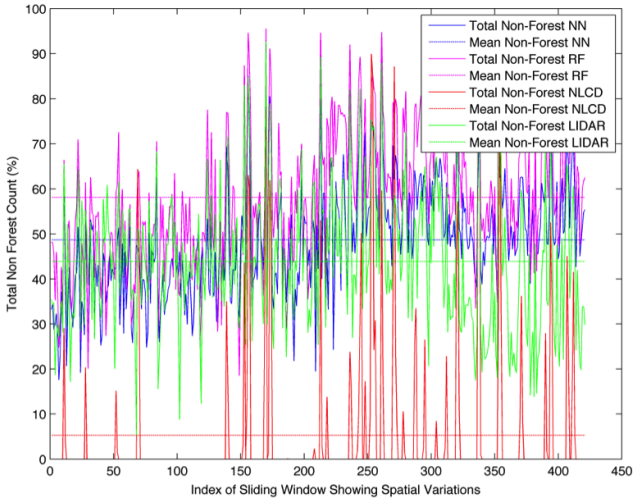
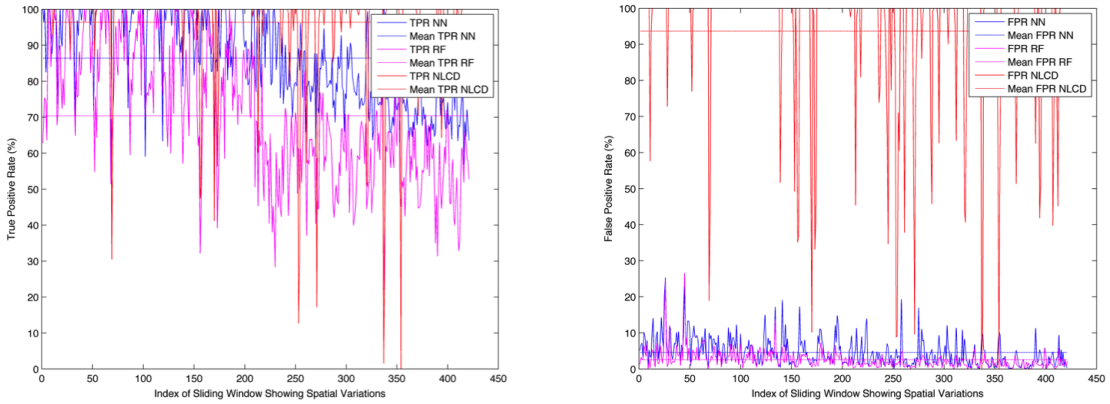


Figure 2.22: Percentage of non-forest area obtained using Neural Network and Random Forest (for NAIP), and NLCD and LiDAR in Area 1 (same as the area in Figure 2.21). The sliding window size was 50×50.



(a) True Positive Rate

(b) False Positive Rate

Figure 2.23: True Positive Rate (TPR) and False Positive Rate (FPR) of Neural Network and Random Forest (for NAIP) and NLCD with LiDAR as ground truth for Area 1 (same as the area in Figure 2.21). The sliding window size was 50×50.

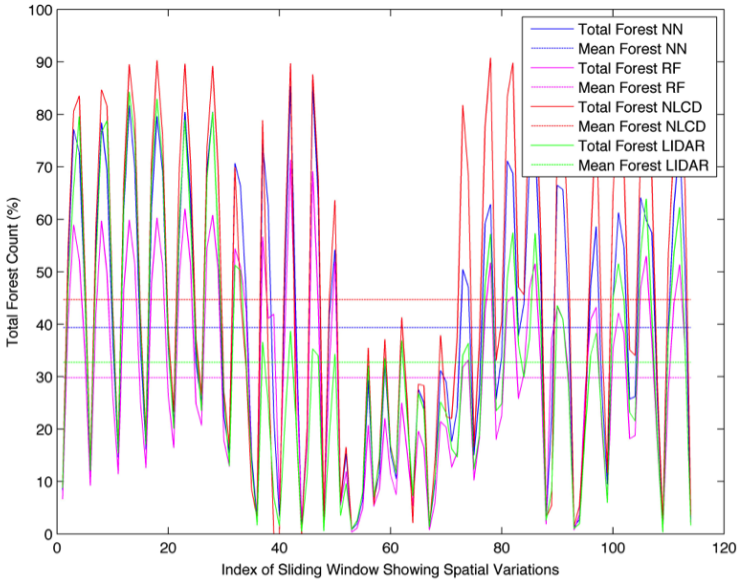


Figure 2.24: Percentage of forest cover obtained using Neural Network and Random Forest (for NAIP), and NLCD and LiDAR for Area 2 (the Chester area in California). The sliding window size was kept as 50×50.

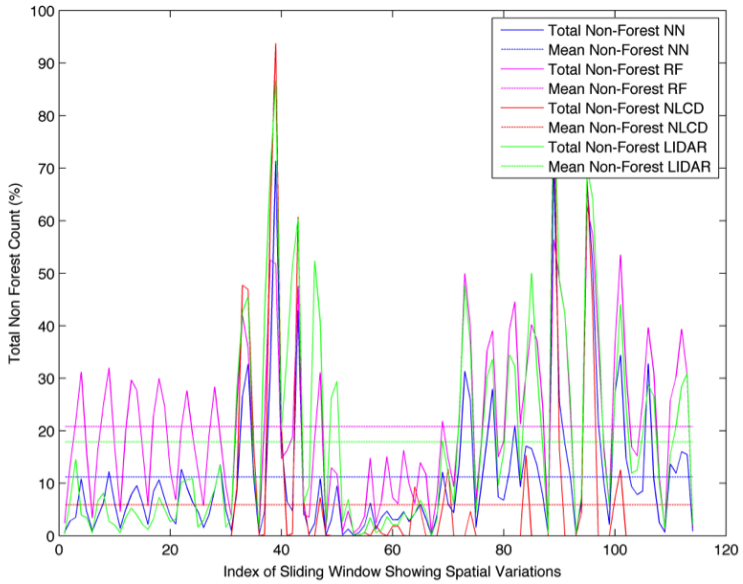
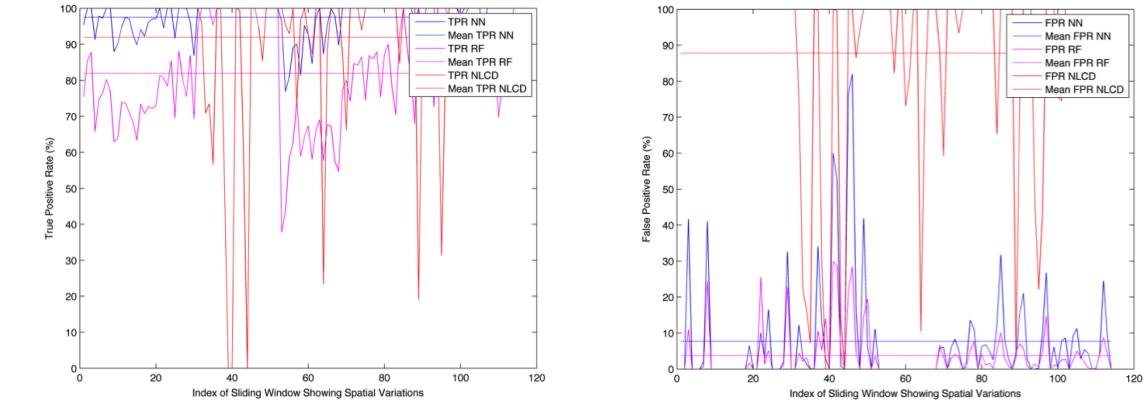


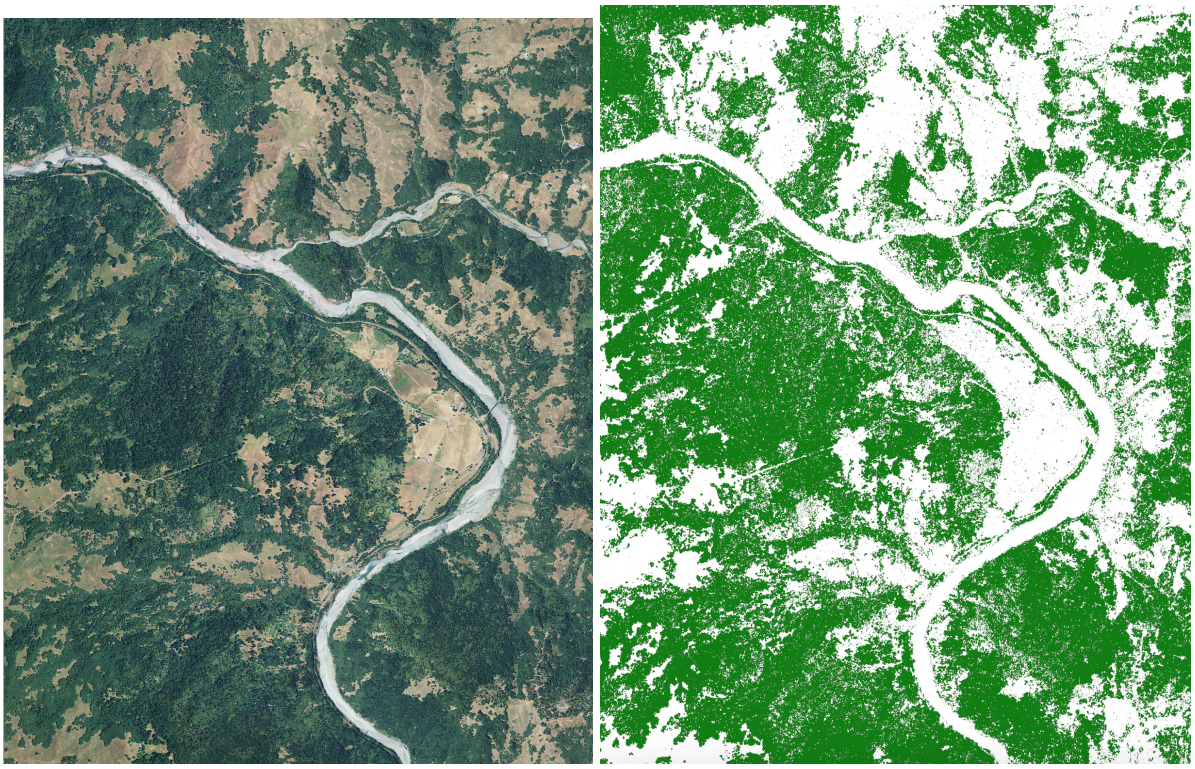
Figure 2.25: Percentage of non-forest area obtained using Neural Network and Random Forest (for NAIP), and NLCD and LiDAR for Area 2 (same as Figure 2.24) with the sliding window size kept constant at 50×50.



(a) True Positive Rate

(b) False Positive Rate

Figure 2.26: True Positive Rate (TPR) and False Positive Rate (FPR) of the Neural Network and Random Forest (for NAIP) and NLCD with LiDAR as ground truth for Area 2 (same as the area in Figure 2.21). The sliding window size was 50×50.



(a) True Positive Rate (b) False Positive Rate

Figure 2.27: A sample NAIP tile (left) and the corresponding binary tree cover mask (right). The white pixels denote non-tree areas while the green pixels denote the tree-cover areas.

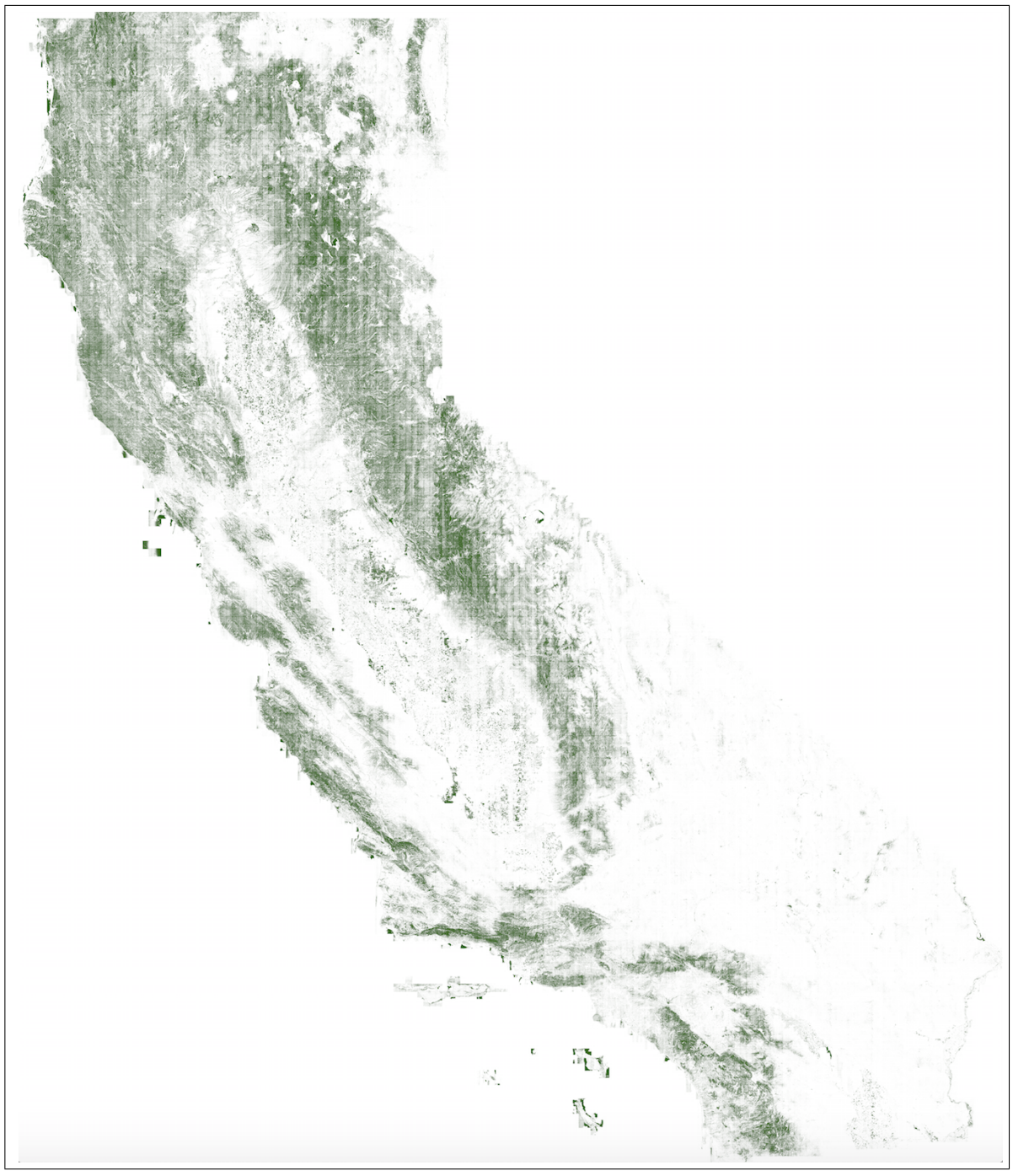


Figure 2.28: The final tree cover map generated by our framework for the whole of California covering 11,095 NAIP tiles. The green pixels denote the tree-cover areas while the white pixels denote the non-tree areas.

Chapter 3
A Deep Learning Approach to Landcover Classification in Aerial Imagery[1]

Satellite image classification is a challenging problem that lies at the crossroads of machine learning, computer vision, and remote sensing. Due to the high variability inherent in satellite data, most of the current object classification approaches are not suitable for handling satellite datasets. The progress of satellite image analytics has also been inhibited by the lack of a single labeled high-resolution dataset with multiple class labels. The contributions of this chapter are twofold – (1) first, we present two new satellite datasets called SAT-4 and SAT-6, and (2) then, we propose a classification framework that extracts features from an input image, normalizes them and feeds the normalized feature vectors to a Deep Belief Network for classification. On the SAT-4 dataset, our best network produces a classification accuracy of 97.95% and outperforms three state-of-the-art object recognition algorithms, namely - Deep Belief Networks, Convolutional Neural Networks and Stacked Autoencoders by $\sim$11%. On SAT-6, it produces a classification accuracy of 93.9% and outperforms the other algorithms by $\sim$15%. Comparative studies with a Random Forest classifier show the advantage of an unsupervised learning approach over traditional supervised learning techniques. A statistical analysis based on Distribution Separability Criterion substantiates the effectiveness of our approach in learning better representations for satellite imagery.

3.1 Introduction

Deep Learning has gained popularity over the last decade due to its ability to learn data representations in an unsupervised manner and generalize to unseen data samples using hierarchical representations. The most recent and best-known *Deep learning model* is the *Deep Belief Network* [48]. Over the last decade, numerous breakthroughs have been made in the field of Deep

[1]Adapted from the paper titled "DeepSat: A Learning Framework for Satellite Imagery" published in the proceedings of the International Conference on Advances in Geographic Information Systems, ACM Sigspatial 2015 [9].

Learning; a notable one being [65], where a locally connected sparse autoencoder was used to detect objects in the ImageNet dataset [31] producing state-of-the-art results. In [77], Deep Belief Networks have been used for modeling acoustic signals and have been shown to outperform traditional approaches using Gaussian Mixture Models for Automatic Speech Recognition (ASR). Another closely related approach, which has gained much traction over the last decade, is the Convolutional Neural Network [67]. This has been shown to outperform Deep Belief Network in classical object recognition tasks like MNIST [105], and CIFAR [61].

A related and equally hard problem is Satellite image classification. It involves terabytes of data and significant variations due to conditions in data acquisition, pre-processing and filtering. Traditional supervised learning methods like Random Forests [21] do not generalize well for such a large-scale learning problem. A novel classification algorithm for detecting roads in Aerial imagery using Deep Neural Networks was proposed in [76]. The problem of detecting various land cover classes in general is a difficult problem considering the significantly higher intra-class variability in land cover types such as trees, grasslands, barren lands, water bodies, etc. as compared to that of roads. Also, in [76], the authors used a window of size 64×64 to derive contextual information. For our general classification problem, a 64×64 window is too big a context covering a total area of 64m$\times$64m. A tree canopy, or a grassy patch can typically be much smaller than this area and hence we are constrained to use a contextual window having a maximum dimension of 28m$\times$28m.

Traditional supervised learning approaches require carefully selected handcrafted features and substantial amounts of labeled data. On the other hand, purely unsupervised approaches are not able to learn the higher order dependencies inherent in the land cover classification problem. So, we propose a combination of handcrafted features that were first used in [46] and an unsupervised learning framework using Deep Belief Network [48] that can learn data representations from large amounts of unlabeled data.

There hasn't been much research in the field of satellite image classification due to a dearth of labeled satellite image datasets. The most well known labeled satellite dataset is the NLCD

2006 [100], which covers the entire globe and provide a spatial resolution of 30m. However, at this resolution, it becomes extremely difficult to distinguish between various landcover types. A high-resolution dataset acquired at a spatial resolution of 1.2m was used in [76]. However, the total area covered by the datasets namely URBAN1 and URBAN2 was ~ 600 square kilometers, which included both training and testing datasets. The labeling was also available only for roads.

Present classification algorithms used for Moderate-resolution Imaging Spectroradiometer (MODIS)(500-m) [35] or Landsat(30-m) based land cover maps like NLCD [100] produce accuracies of 75% and 78% resp. The relatively lower resolution of the datasets makes it difficult to analyze the performance of these algorithms for 1-m imagery. A method based on object detection using Bayes framework and subsequent clustering of the objects using Latent Dirichlet Allocation was proposed in [96]. However, their approach detects object groups at a higher level of abstraction like parking lots. Detecting the objects like cars or trees in itself is not addressed in their work. A deep convolutional hierarchical framework was proposed recently by [84]. However, they report results on the AVIRIS Indiana's Indian Pines test site. The spatial resolution of the dataset is limited to 20m and it is difficult to evaluate the performance of their algorithm for object recognition tasks at a higher resolution. An evaluation of various feature learning strategies was done in [93]. They evaluated both feature extraction techniques as well as classifiers like DBN and Random Forest for various aerial datasets. However, since the training data was significantly limited, the DBN was not able to produce any improvements over Random Forest even when raw pixel values were fed into the classifier. In contrast, our study shows that DBNs can be better classifiers when there is significant amount of training data to initialize the neural network at a global error basin.

The main contributions of our work are twofold – (1) We first present two labeled datasets of satellite images – SAT-4 and SAT-6 covering a total area of ~ 800 square kilometers, which can be used to further the research and investigate the use of various learning models for satellite image classification. Both SAT-4 and SAT-6 are sampled from a much larger dataset [108], which covers the whole of continental United States and can be used to create labeled landcover maps,

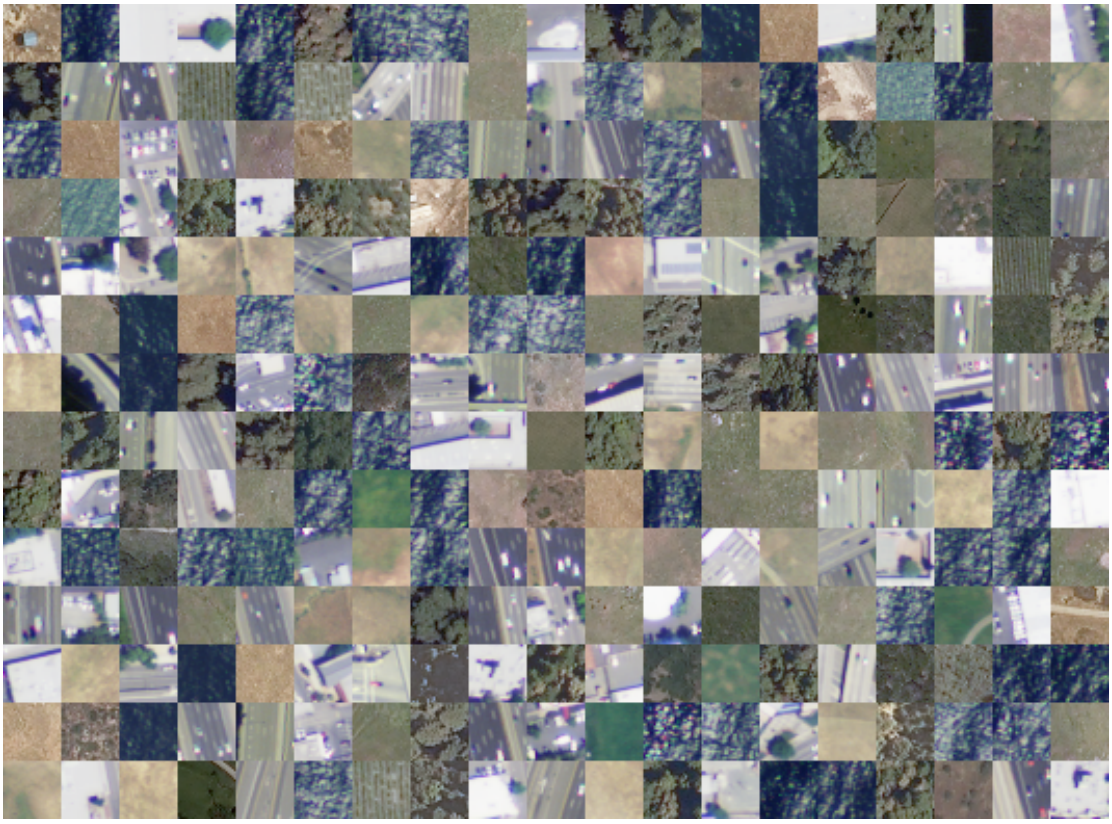


Figure 3.1: Sample images from the SAT-6 dataset

which can then be used for various applications such as measuring ground carbon content or estimating total area of rooftops for solar power generation.

(2) Next, we present a framework for the classification of satellite imagery that a) extracts features from the image, b) normalizes the features, and c) feeds the normalized feature vectors to a Deep Belief Network for classification. On the SAT-4 dataset, our framework outperforms three state-of-the-art object recognition algorithms - Deep Belief Networks, Convolutional Neural Networks and Stacked Autoencoders by $\sim$11% and produces an accuracy of 97.95%. On SAT-6, it produces an accuracy of 93.9% and outperforms the other algorithms by $\sim$15%. We also present a statistical analysis based on Distribution Separability Criterion to justify the effectiveness of our feature extraction approach to obtain better representations for satellite data.

3.2 Dataset[2]

Images were extracted from the National Agriculture Imagery Program (NAIP [108]) dataset. The NAIP dataset consists of a total of 330,000 scenes spanning the whole of the Continental United States (CONUS). We used the uncompressed digital Ortho quarter quad tiles (DOQQs) which are GeoTIFF images and the area corresponds to the United States Geological Survey (USGS) topographic quadrangles. The average image tiles are $\sim$6000 pixels in width and $\sim$7000 pixels in height, measuring around 200 megabytes each. The entire NAIP dataset for CONUS is $\sim$65 terabytes. The imagery is acquired at a 1-m ground sample distance (GSD) with a horizontal accuracy that lies within six meters of photo-identifiable ground control points [106]. The images consist of 4 bands – red, green, blue and Near Infrared (NIR). In order to maintain the high variance inherent in the entire NAIP dataset, we sample image patches from a multitude of scenes (a total of 1500 image tiles) covering different landscapes like rural areas, urban areas, densely forested, mountainous terrain, small to large water bodies, agricultural areas, etc. covering the whole state of California. An image labeling tool developed as part of this study was used to manually label uniform image patches belonging to a particular landcover class. Once labeled, 28×28 non-overlapping sliding window blocks were extracted from the uniform image patch and saved to the dataset with the corresponding label. We chose 28×28 as the window size to maintain a significantly bigger context as pointed by [76], and at the same time not to make it as big as to drop the relative statistical properties of the target class conditional distributions within the contextual window. Care was taken to avoid interclass overlaps within a selected and labeled image patch. Sample images from the dataset are shown in Figure 3.1.

3.2.1 SAT-4

SAT-4 consists of a total of 500,000 image patches covering four broad land cover classes. These include – barren land, trees, grassland and a class that consists of all land cover classes other than the above three. 400,000 patches (comprising of four-fifths of the total dataset) were chosen for training and the remaining 100,000 (one-fifths) were chosen as the testing dataset. We

[2]The full datasets are available at the web link [102]

ensured that the training and test datasets belong to disjoint set of image tiles. Each image patch is size normalized to 28×28 pixels. Once generated, both the training and testing datasets were randomized using a pseudo-random number generator.

3.2.2 SAT-6

SAT-6 consists of a total of 405,000 image patches each of size 28×28 and covering 6 land-cover classes – barren land, trees, grassland, roads, buildings and water bodies. 324,000 images (comprising of four-fifths of the total dataset) were chosen as the training dataset and 81,000 (one fifths) were chosen as the testing dataset. Similar to SAT-4, the training and test sets were selected from disjoint NAIP tiles. Once generated, the images in the dataset were randomized in the same way as that for SAT-4. The specifications for the various landcover classes of SAT-4 and SAT-6 were adopted from those used in the National Land Cover Data (NLCD) algorithm [109].

3.3 Investigation of various Deep Learning Models

3.3.1 Deep Belief Network

Deep Belief Network (DBN) consists of multiple layers of stochastic, latent variables trained using an unsupervised learning algorithm followed by a supervised learning phase using feed-forward backpropagation Neural Networks. In the unsupervised pre-training stage, each layer is trained using a Restricted Boltzmann Machine (RBM). Unsupervised pre-training is an important step in solving a classification problem with terabytes of data and high variability. A DBN is a graphical model [59] where neurons of the hidden layer are conditionally independent of each other given a particular configuration of the visible layer and vice versa. A DBN can be trained layer-wise by iteratively maximizing the conditional probability of the input vectors or visible vectors given the hidden vectors and a particular set of layer weights. As shown in [48], this layer-wise training can help in improving the variational lower bound on the probability of the input training data, which in turn leads to an improvement of the overall generative model.

A RBM is trained using a *Contrastive Divergence* algorithm [23]. Once trained, the DBN

can be used to initialize the weights of the Neural Network for the supervised learning phase [13].

Next, we investigate the classification accuracy of various architectures of DBN on both SAT-4 and SAT-6 datasets.

- **DBN Results on SAT-4 & SAT-6**

To investigate the performance of the DBN, we experiment with both *big* and *deep* neural architectures. This is done by varying the number of neurons per layer as well as the total number of layers in the network. Our objective is to investigate whether the more complex features learned in the deeper layers of the DBN are able to provide the network with the discriminative power required to handle higher-order texture features typical of satellite imagery data. The results from the DBN for various network architectures for SAT-4 and SAT-6 are enumerated in Table 3.1. Each network was trained for a maximum of 500 epochs and the network state with the lowest validation error was used for testing. It can be seen from the table that for both SAT-4 and SAT-6, the classifier accuracy initially improves and then falls as more neurons or layers are added to the network.

Network Arch. Neurons/layer [Layers]	**Classifier Accuracy SAT-4 (%)**	**Classifier Accuracy SAT-6 (%)**
100 [2]	79.74	68.51
100 [3]	**81.78**	**76.47**
100 [4]	79.802	74.44
100 [5]	62.776	63.14
500 [2]	68.916	60.35
500 [3]	71.674	61.12
500 [4]	65.002	57.31
500 [5]	64.174	55.78

Table 3.1: Classification Accuracy of DBN with various architectures on SAT-4 and SAT-6

3.3.2 Convolutional Neural Network

Convolutional Neural Network (CNN) first introduced in [36] is a hierarchical model inspired by the human visual cortical system [52]. It was significantly improved and applied to document recognition in [67]. A committee of 35 convolutional neural nets with elastic distortions and width normalization [27] has produced state-of-the-art results on the MNIST handwritten dig-

its dataset. CNN consists of a hierarchical representation using convolutional layers and fully connected layers, with non-linear transformations and feature pooling.

We investigate the use of different CNN architectures for SAT-4 and SAT-6 as detailed below.

- **CNN Results on SAT-4 & SAT-6**

For CNN, we vary the number of feature maps in each layer as well as the total number of convolutional and subsampling layers. The results from various network configurations with increasing number of maps and layers is enumerated in Table 3.2. For the experiments, we used both 3×3 and 5×5 kernels for the convolutional layers and 3×3 averaging and max-pooling kernels for the sub-sampling layers. We also use overlapping pooling windows with a stride size of 2 pixels. The last sub-sampling layer is connected to a fully-connected layer with 64 neurons. The output of the fully-connected layer is fed into a 4-way softmax function that generates a probability distribution over the 4 class labels of SAT-4 and a 6-way softmax for the 6 class labels of SAT-6. In Table 3.2, the "Ac-Bs(n)" notation denotes that the network has a convolutional layer with A feature maps followed by a sub-sampling layer with a kernel of size $B\times B$. 'n' denotes the type of pooling function in the sub-sampling layer, 'a' denotes average pooling while 'm' denotes max-pooling. From the table, it can be seen that the smallest networks consistently produce the best results. Also, both for SAT-4 and SAT-6, using networks with convolution kernels of size 3×3 leads to a significant drop in classifier accuracy. The biggest networks with 50 maps per layer also exhibit significant drop in classifier accuracy.

Network Architecture (Convolution kernel size)	**Accuracy SAT-4 (%)**	**Accuracy SAT-6 (%)**
6c-3s(a)-12c-3s(m) (5×5)	**86.827**	**79.063**
18c-3s(a)-36c-3s(m) (5×5)	82.325	78.704
6c-3s(a)-12c-3s(m)-12c -3s(m)(5×5)	81.907	76.963
50c-3s(a)-50c-3s(m)-50c -3s(m)(5×5)	73.85	75.689
6c-3s(a)-12c-3s(m) (3×3)	73.811	54.385
6c-3s(m)-12c-3s(m) (5×5)	85.612	77.636

Table 3.2: Classification Accuracy of CNN with various architectures on SAT-4

3.3.3 Stacked Autoencoder

A Stacked Autoencoder (SAE) [98] consists of a combination of multiple sparse autoencoders, which can be trained in a greedy-layerwise fashion similar to that of Restricted Boltzmann Machines in a DBN. Each autoencoder is associated with a set of weights and biases. In the SAE, each layer can be trained independent of the other layers. Once trained, the parameters of an autoencoder are frozen in place. The training algorithm consists of two passes – a forward pass and a backward pass. The forward pass, also called as the encoding phase encodes raw image pixels into an increasingly higher-order representation. The backward pass simply performs the reverse operation by decoding these higher-order features into simpler representations.

The hidden unit activations of the neurons in the deepest layer are used for classification after a supervised fine-tuning using backpropagation.

- **SAE Results on SAT-4 & SAT-6**

Different network configurations were chosen for the SAE in a manner similar to that described above for DBN and CNN. The results are enumerated in Table 3.3. Similar to DBN, each network is trained for a maximum of 500 epochs and the lowest test error is considered for evaluation. As highlighted in the Table, networks with 5 layers and 100 neurons in each layer produce the best results on both SAT-4 and SAT-6. It can be seen from the table that on both datasets, the classifier accuracy initially improves and then drops with increasing number of neurons and layers, similar to that of DBN. Also, the biggest networks with 500 and 2352 neurons in each layer exhibit a significant drop in classifier accuracy.

3.4 DeepSat - A Detailed Architectural Overview

Figure 3.2 schematically describes our proposed classification framework. Instead of the traditional DBN model described in Section 4.4, which takes as input the multi-channel image pixels reshaped as a linear vector, our classification framework first extracts features from the image which in turn are fed as input to the DBN after normalizing the feature vectors.

Network Arch. Neurons/layer [Layers]	Classifier Accuracy SAT-4 (%)	Classifier Accuracy SAT-6 (%)
100 [1]	75.88	74.89
100 [2]	76.854	76.12
100 [3]	77.804	76.45
100 [4]	78.674	76.52
100 [5]	**79.978**	**78.43**
100 [6]	75.766	76.72
500 [3]	63.832	54.37
2352 [2]	51.766	37.121

Table 3.3: Classification Accuracy of SAE with various architectures on SAT-4 and SAT-6

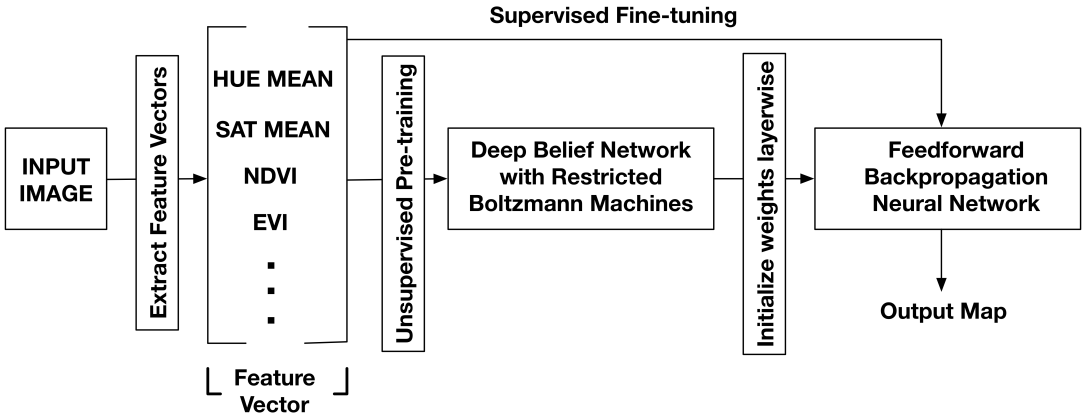


Figure 3.2: Schematic of the classification framework

3.4.1 Feature Extraction

The feature extraction phase computes 150 features from the input imagery. The key features that we use for classification are mean, standard deviation, variance, 2nd moment, direct cosine transforms, correlation, co-variance, autocorrelation, energy, entropy, homogeneity, contrast, maximum probability and sum of variance of the hue, saturation, intensity, and NIR channels as well as those of the color co-occurrence matrices. These features were shown to be useful descriptors for classification of satellite imagery in previous studies ([46], [57], [28]). Since two of the classes in SAT-4 and SAT-6 are trees and grasslands, we incorporate features that are useful determinants for segregation of vegetated areas from non-vegetated ones. The red band already provides a useful feature for discrimination of vegetated and non-vegetated areas based on

chlorophyll reflectance, however, we also use derived features (vegetation indices derived from spectral band combinations) that are more representative of vegetation greenness - this includes the Enhanced Vegetation Index (EVI [53]), Normalized Difference Vegetation Index (NDVI [85], [95]) and Atmospherically Resistant Vegetation Index (ARVI [56]).

These indices are expressed as follows:

$$EVI = G \times \frac{NIR - Red}{NIR + c_{red} \times Red - c_{blue} \times Blue + L} \tag{3.1}$$

Here, the coefficients G, c_{red}, c_{blue} and L are chosen to be 2.5, 6, 7.5 and 1 following those adopted in the MODIS EVI algorithm [106].

$$NDVI = \frac{NIR - Red}{NIR + Red} \tag{3.2}$$

$$ARVI = \frac{NIR - (2 \times Red - Blue)}{NIR + (2 \times Red + Blue)} \tag{3.3}$$

The performance of our learner depends to a large extent on the selected features. Some features contribute more than others towards optimal classification. The 150 features extracted are narrowed down to 22 using a feature-ranking algorithm based on Distribution Separability Criterion [19]. Details of the feature ranking method along with the ranking for all the 22 features used in our framework is listed in Section 3.6.1.

3.4.2 Data Normalization

The feature vectors extracted from the training and test datasets are separately normalized to lie in the range $[0, 1]$. This is done using the following equation:

$$F_{normalized} = \frac{F - F_{min}}{F_{max} - F_{min}} \tag{3.4}$$

where, F_{min} and F_{max} are computed for a particular feature type over all images in the dataset.

3.4.3 Classification

The set of normalized feature descriptors extracted from the input image is fed into the DBN, which is then trained using *Contrastive divergence* in the same way as explained in Section 4.4. Once trained the DBN is used to initialize the weights of a feedforward backpropagation neural network.

The neural network gives an estimate of the posterior probabilities of the class labels, given the input vectors, which is the feature vector in our case. As illustrated in [18], the outputs of a neural network trained by minimizing the sum of squares error function approximates the conditional averages of the target data

$$y_k(x) = \langle t_k|x\rangle = \int t_k p(t_k|x)dt_k \tag{3.5}$$

Here, t_k are the set of target values that represent the class membership of the input vector x_k. For a binary classification problem, in order to map the outputs of the neural network to the posterior probabilities of the labeling, we use a single output y and a target coding that sets $t^n = 1$ if x^n is from class C_1 and $t^n = 0$ if x^n is from class C_2. The target distribution would then be given as

$$p(t_k|x) = \delta(t-1)P(C_1|x) + \delta(t)P(C_2|x) \tag{3.6}$$

Here, δ represents the Dirac delta function. This function exhibits the property $\delta(x) = 0$ if $x \neq 0$ and

$$\int_{-\infty}^{\infty} \delta(x)\,\mathrm{d}x = 1 \tag{3.7}$$

From 4.8 and 4.9, we get

$$y(x) = P(C_1|x) \tag{3.8}$$

So, the network output $y(x)$ represents the posterior probability of the input vector x having the class membership C_1 and the probability of the class membership C_2 is given by $P(C_2|x) = 1 - y(x)$. This argument can easily be extended to multiple class labels for a generalized multi-class classification problem.

The feature extraction phase proves to be a useful dimensionality reduction technique that helps improve the discriminative power of the DBN based classifier significantly.

3.5 Results and Comparative Studies

The feature vectors extracted from the dataset are fed into DBNs with different configurations. Since, the feature vectors create a low dimensional representation of the data, so, DeepSat converges to high accuracy even with a much smaller network with fewer layers and very few neurons per layer. This speeds up network training by several orders of magnitude.

The models based on Deep Belief Networks and Stacked Autoencoders were implemented in Matlab while the Convolutional Neural Network models were implemented using the Caffe Deep Learning framework [55].

Various network architectures along with the classification accuracy for DeepSat on the SAT-4 and SAT-6 datasets are listed in Table 3.4. From the Table, it is evident that the best performing DeepSat network outperforms the best traditional Deep Learning approach (CNN) by $\sim$11% on the SAT-4 dataset and by $\sim$15% on the SAT-6 dataset. Table 3.5 shows the training and test times of a traditional DBN and DeepSat on SAT-4 and SAT-6.

We also compare DeepSat with a Random Forest classifier to investigate the advantages gained by unsupervised pre-training in DBN as opposed to the traditional supervised learning in Random Forests. On SAT-4, the Random forest classifier produces an accuracy of 69% while on SAT-6, it produces an accuracy of 54%. The highest accuracy was obtained for a forest with 100 trees. Further increase in the number of trees did not yield any significant improvement in classifier accuracy. It can be easily seen that the various Deep architectures produce better classification accuracy than the Random Forest classifier which relies solely on supervised learning.

Network Arch. Neurons/layer [Layers]	Classifier Accuracy SAT-4 (%)	Classifier Accuracy SAT-6 (%)
10 [2]	96.585	91.91
10 [3]	96.8	87.716
20 [2]	97.115	86.21
20 [3]	95.473	93.42
50 [2]	**97.946**	**93.916**
50 [3]	97.654	92.65
100 [2]	97.292	89.08
100 [3]	95.609	91.057

Table 3.4: Classification Accuracy of DeepSat with various network architectures on SAT-4 and SAT-6

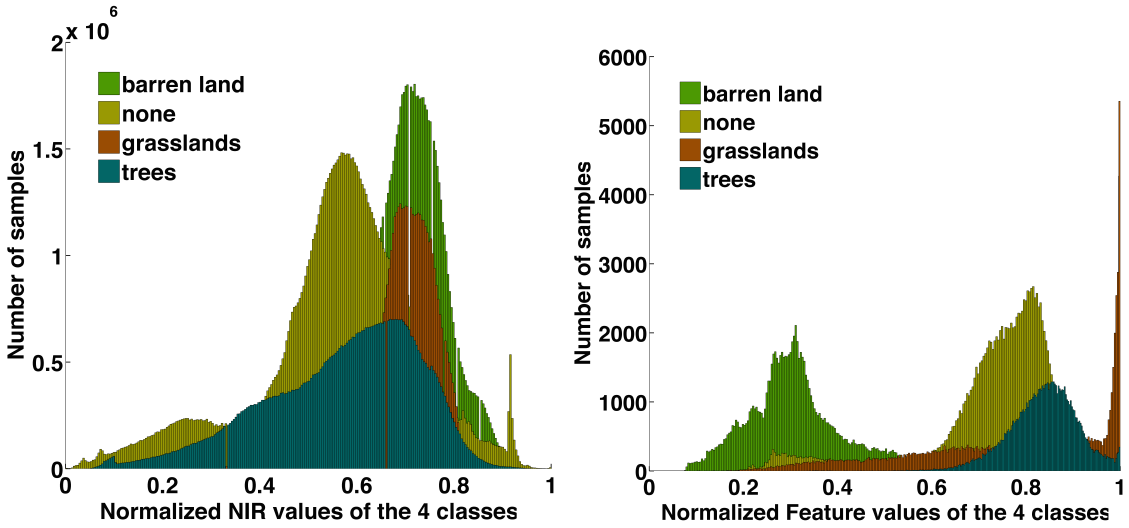
Dataset	DBN Train Time	DeepSat Train Time	DBN Test Time	DeepSat Test Time
SAT-4	120.68 s	78.19 s	4.71 s	2.31 s
SAT-6	112.75 s	73.28 s	3.70 s	1.96 s

Table 3.5: Training and Test times of a traditional DBN and DeepSat on SAT-4 and SAT-6

3.6 Why Traditional Deep Architectures are not enough for SAT-4 & SAT-6?

While traditional Deep Learning approaches have produced state-of-the-art results for various pattern recognition problems like handwritten digit recognition [105], object recognition [61], face recognition [92], etc., but satellite datasets have high intra and inter-class variability and the amount of labeled data is much smaller as compared to the total size of the dataset. Also, higher-order texture features are a very important discriminative parameter for various landcover classes. On the contrary, shape/edge based features which are predominantly learned by various Deep architectures are not very useful in learning data representations for satellite imagery. This explains the fact why traditional Deep architectures are not able to converge to the global optima even for reasonably large as well as Deep architectures.

Also, spatially contextual information is another important parameter for modeling satellite imagery. In traditional Deep Learning approaches like DBN and SAE, the relative spatial information of the pixels is lost. As a result the orderless pool of pixel values which acts as input



(a) Distribution of NIR on the SAT-4 classes

(b) Distribution of a sample DeepSat feature (Auto-correlation of Hue Color co-occurance matrix) on the SAT-4 classes

Figure 3.3: Distributions of the raw NIR values for traditional Deep Learning Algorithms and a sample DeepSat feature for various classes on SAT-4 (*Best viewed in color*)

to the Deep Networks lack sufficient discriminative power to be well-represented even by very big and/or deep networks. CNN however, involves feature-pooling from a local spatial neighborhood, which justifies its improved performance over the other two algorithms on both SAT-4 and SAT-6. Even though our approach extracts an orderless pool of feature vectors, the spatial context is already well-represented in the individual feature values themselves. We substantiate our arguments about the effectiveness of our feature extraction approach from a statistical point of view as detailed in the analysis below.

		Dist. b/w Means	**Standard Deviations**
SAT-4	Raw Images	0.1994	0.1166
	DeepSat Features	**0.8454**	**0.0435**
SAT-6	Raw Images	0.3247	0.1273
	DeepSat Features	**0.9726**	**0.0491**

Table 3.6: Distance between Means and Standard Deviations for raw image values and DeepSat feature vectors for SAT-4 and SAT-6

3.6.1 A Statistical Perspective based on Distribution Separability Criterion

Improving classification accuracy can be viewed as maximizing the separability between the class-conditional distributions. Following the analysis presented in [19], we can view the problem of maximizing distribution separability as maximizing the distance between distribution means and minimizing their standard deviations. Figure 3.3 shows the histograms that represent the class-conditional distributions of the NIR channel and a sample feature extracted in the DeepSat framework. As illustrated in Table 3.6, the features extracted in DeepSat have a higher distance between means and a lower standard deviation as compared to the original image distributions, thereby ensuring better class separability.

- **Feature Ranking**

Following the analysis proposed in Section 3.6.1 above, we can derive a metric for the Distribution Separability Criterion as follows:

$$D_s = \frac{\overline{\|\delta_{mean}\|}}{\overline{\delta_\sigma}} \tag{3.9}$$

where $\overline{\|\delta_{mean}\|}$ indicates the mean of distance between means and $\overline{\delta_\sigma}$ indicates the mean of standard deviations of the class conditional distributions. Maximizing D_s over the feature space, a feature ranking can be obtained. Table 3.7 shows the ranking of the various features used in our framework along with the values of the corresponding distance between means $\overline{\|\delta_{mean}\|}$, standard deviation $\overline{\delta_\sigma}$ and Distribution Separability Criterion D_s.

- **Distribution Separability and Classifier Accuracy**

In order to analyze the improvements achieved in the learning framework due to the feature extraction step, we measured the Distribution Separability of the mean activation of the neurons in each layer of the DBN and that of DeepSat. The results are noted in Figure 3.4. It can be seen that the mean activation learned by each layer of DeepSat exhibit a significantly higher distribution separability (by several orders of magnitude) than the neurons of a DBN. This justifies the significant improvement in performance of DeepSat (using the features) as compared to the DBN based framework (using the raw pixel values as input). Also, a comparison of Figure 3.4 with

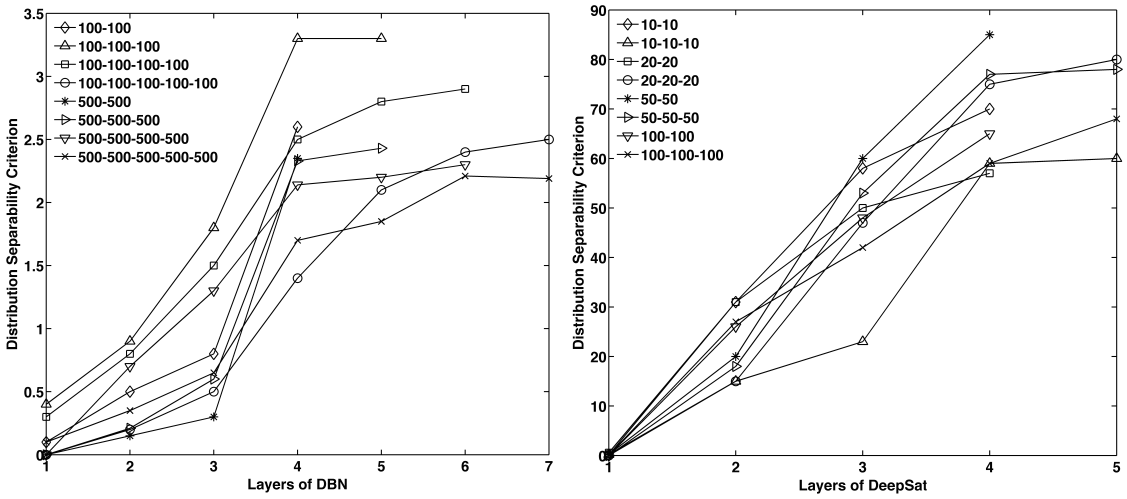
Rank	Feature	$\overline{\|\delta_{mean}\|}$	$\overline{\delta_{\sigma}}$	D_s
1	I CCM mean	0.4031	0.1371	2.9403
2	H CCM sosvh	0.2359	0.0928	2.5413
3	H CCM autoc	0.2334	0.1090	2.1417
4	S CCM mean	0.0952	0.0675	1.4099
5	H CCM mean	0.0629	0.0560	1.1237
6	SR	0.0403	0.0428	0.9424
7	S CCM 2nd moment	0.0260	0.0312	0.8354
8	I CCM 2nd moment	0.0260	0.0312	0.8354
9	I 2nd moment	0.0260	0.0312	0.8345
10	I variance	0.0260	0.0312	0.8345
11	NIR std	0.0251	0.0315	0.7980
12	I std	0.0251	0.0314	0.7968
13	H std	0.0252	0.0317	0.7956
14	H mean	0.0240	0.0314	0.7632
15	I mean	0.0254	0.0336	0.7541
16	S mean	0.0232	0.0319	0.7268
17	I CCM covariance	0.0378	0.0522	0.7228
18	NIR mean	0.0246	0.0351	0.6997
19	ARVI	0.0229	0.0345	0.6622
20	NDVI	0.0215	0.0326	0.6594
21	DCT	0.0344	0.0594	0.5792
22	EVI	0.0144	0.0450	0.3207

Table 3.7: Ranking of features based on Distribution Separability Criterion for SAT-6

Table 3.1 and Table 3.4 shows that the distribution separabilities using the various architectures of the DBN and DeepSat are positively correlated to the final classifier accuracy. This justifies the effectiveness of our distribution separability metric D_s as a measure of the final classifier accuracy.

3.7 What is the difference between MNIST, CIFAR-10 and SAT-6 in terms of dimensionality?

We argue that handwritten digit datasets like MNIST and object recognition datasets like CIFAR-10 lie on a much lower dimensional manifold than the airborne SAT-6 dataset. Hence,



(a) Distribution Separability Criterion of DBN (b) Distribution Separability Criterion of DeepSat

Figure 3.4: Distribution Separability Criterion of the neurons in the layers of a DBN and DeepSat with various architectures on SAT-6

even if Deep Neural Networks can effectively classify the raw feature space of object recognition datasets but the dimensionality of the airborne image datasets is such that Deep Neural Networks cannot classify them. In order to estimate the dimensionality of the datasets, we use the concept of *intrinsic dimension*[24].

3.7.1 Intrinsic Dimension Estimation using the DanCo algorithm

To estimate the intrinsic dimension of a dataset, we use the DANCo algorithm [24]. It uses the complementary information provided by the normalized nearest neighbor distances and angles calculated on pairs of neighboring points.

Taking 10 rounds of 1000 random samples and averaging, we obtain the intrinsic dimension for the MNIST, CIFAR-10 and SAT-6 datasets and the Haralick features extracted from the SAT-6 dataset. The results are listed in Table 3.8.

Dataset	**Intrinsic Dimension**
MNIST	16
CIFAR-10	17
SAT-6	115
Haralick Features extracted from SAT-6	**4.2**

Table 3.8: Intrinsic Dimension estimation using DANCo on the MNIST, CIFAR-10, and SAT-6 datasets and the Haralick features extracted from the SAT-6 dataset.

So, it can be seen that the intrinsic dimensionality of the SAT-6 dataset is orders of magnitude higher than that of MNIST. So, a deep neural network finds it difficult to classify the SAT-6 dataset because of its intrinsically high dimensionality. However, as seen in the equation above, the features extracted from SAT-6 have a much lower intrinsic dimensionality and lie on a much lower dimensional manifold than the raw vectors and hence can be classified even by networks with relatively smaller architectures.

3.7.2 Visualizing Data in an n-dimensional space

We can visualize the data as distributed in an n-dimensional unit hypersphere

Volume of the sphere,

$$V_{sphere} = \frac{\pi^{\frac{n}{2}}}{\Gamma(\frac{n}{2}+1)} R^n = \frac{\pi^{\frac{n}{2}}}{\Gamma(\frac{n}{2}+1)} \tag{3.10}$$

for n-dimensional Euclidean space and Γ is Euler's gamma function. Now, the total volume of the n-dimensional space can be accounted by the volume of an n-dimensional hypercube of length 2 embedding the hypersphere, i.e, Volume of the n-cube,

$$V_{cube} = R^n = 2^n \tag{3.11}$$

So, the relative fraction of the data points which lie on the sphere as compared to the data points on the n-dimensional embedding space is given as

$$V_{relative} = \frac{V_{sphere}}{V_{cube}} = \frac{\pi^{\frac{n}{2}}}{2^n \Gamma(\frac{n}{2}+1)} \tag{3.12}$$

$$V_{relative} \to 0 \text{ as } n \to \infty \tag{3.13}$$

This means that as the dimensionality of sample data approaches ∞, the spread or scatter of the data points approaches 0 with respect to the total search space. As a result, various classification and clustering algorithms lose their discriminative power in higher dimensional feature spaces.

3.8 Discussion

The semi-supervised learning framework proposed in this chapter significantly outperforms the state-of-the-art by $\sim$11% and $\sim$15% respectively. The feature extraction phase computes some features derived from the remote sensing literature and significantly improves the discriminative power of the learning framework. For satellite datasets, with inherently high variability, traditional deep learning approaches are unable to converge to a global optima even with significantly big and deep architectures. A statistical analysis based on Distribution Separability Criterion justifies the effectiveness of the feature extraction approach in generating optimal discriminative representations for aerial imagery datasets.

Chapter 4
Learning Sparse Feature Representations using Probabilistic Quadtrees and Deep Belief Nets[1]

Learning sparse feature representations is a useful instrument for solving an unsupervised learning problem. In this paper, we present three labeled handwritten digit datasets, collectively called n-MNIST by adding noise to the MNIST dataset and three labeled datasets formed by adding noise to the offline Bangla numeral database. Then, we propose a novel framework for the classification of handwritten digits that learns sparse representations using probabilistic quadtrees and Deep Belief Nets. On the MNIST, n-MNIST and noisy Bangla datasets, our framework shows promising results and significantly outperforms traditional Deep Belief Networks.

4.1 Introduction

Deep Learning has gained popularity over the last decade due to its ability to learn data representations in an unsupervised manner and generalize to unseen data samples using hierarchical representations. One of the popular unsupervised models in Deep learning is the *Deep Belief Network*[48]. In [77], Deep Belief Networks have been used for modeling acoustic signals and have been shown to outperform traditional approaches using Gaussian Mixture Models for Automatic Speech Recognition (ASR). Deep Belief Network is trained one layer at a time. The layers are trained separately using a greedy layerwise pre-training using a Restricted Boltzmann Machine (RBM). A sparse feature learning algorithm for Deep Belief Networks was proposed in [83]. However, their work was focused on maximization of information content in the learned representations. Restricted Boltzmann Machines, on the other hand, are trained by minimizing a contrastive term in the loss function.

The main contributions of our work are twofold – (1) We first present three labeled hand-

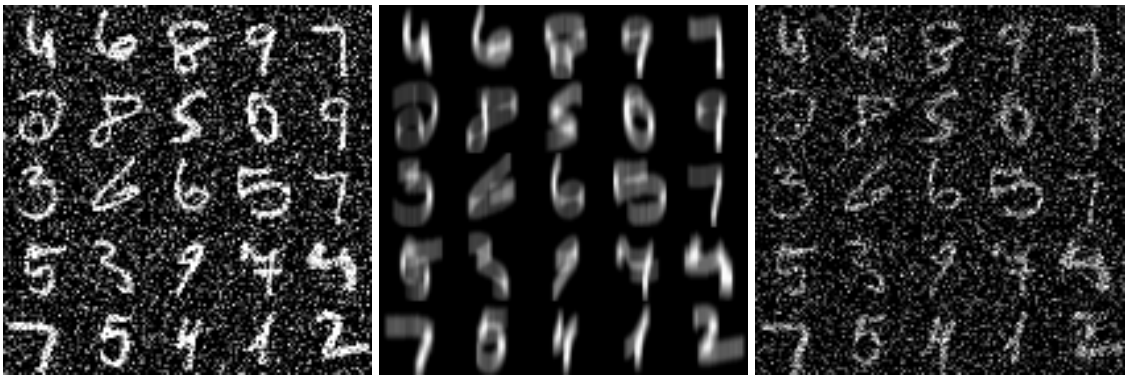
[1] Adapted from the paper titled "Learning Sparse Feature Representations using Probabilistic Quadtrees and Deep Belief Nets" published at the proceedings of the European Symposium on Artificial Neural Networks, ESANN 2015 [12].

written digit datasets, collectively called n-MNIST and three labeled Bangla numeral datasets, created by adding white gaussian noise, adding a motion blur and by reducing the contrast of the original MNIST dataset[105] and the offline handwritten Bangla numeral dataset [16]. (2) Then, we present a framework for the classification of handwritten digits that a) learns probabilistic quadtrees from the dataset, b) performs a Depth First Search on the quadtrees to create sparse representations in the form of linear vectors, and c) feeds the linear vectors into a Deep Belief Network for classification. On the MNIST, n-MNIST and noisy Bangla datasets, our framework shows promising results and significantly outperforms traditional Deep Belief Networks.

4.2 Datasets[2]

We evaluate our framework on the MNIST dataset[105] of handwritten digits as well as three artificial datasets collectively called n-MNIST (noisy MNIST) created by adding – (1) additive white gaussian noise, (2) motion blur and (3) a combination of additive white gaussian noise and reduced contrast to the MNIST dataset. Another set of experiments were performed on the offline Bangla numeral database [16]. The additive white gaussian noise was created by adding a noise which has a signal to noise ratio (SNR) of 9.5. The Motion Blur filter is made to emulate a motion of the camera by τ pixels linearly, at an angle of θ degrees. The filter becomes a vector for horizontal and vertical motions. We use a value of 5 pixels for τ and a value of 15 degrees in the counterclockwise direction is chosen for θ. For the noisy dataset with reduced contrast and AWGN, the contrast range was scaled down to half and was applied with an Additive White Gaussian Noise having a signal to noise ratio (SNR) of 12. This emulates background clutter along with significant change in lighting conditions. Some of the images from the noisy MNIST dataset are presented in Figure 4.1 and those from the noisy Bangla dataset are presented in Figure 4.2.

[2]The datasets are available at the web link [107] and [110] along with a detailed description of the methods and parameters used to create them



(a) MNIST with Additive White Gaussian Noise

(b) MNIST with Motion Blur

(c) MNIST with AWGN and reduced contrast

Figure 4.1: Example images from the n-MNIST dataset created as part of the experiments.

4.2.1 Pre-processing for the Bangla Dataset

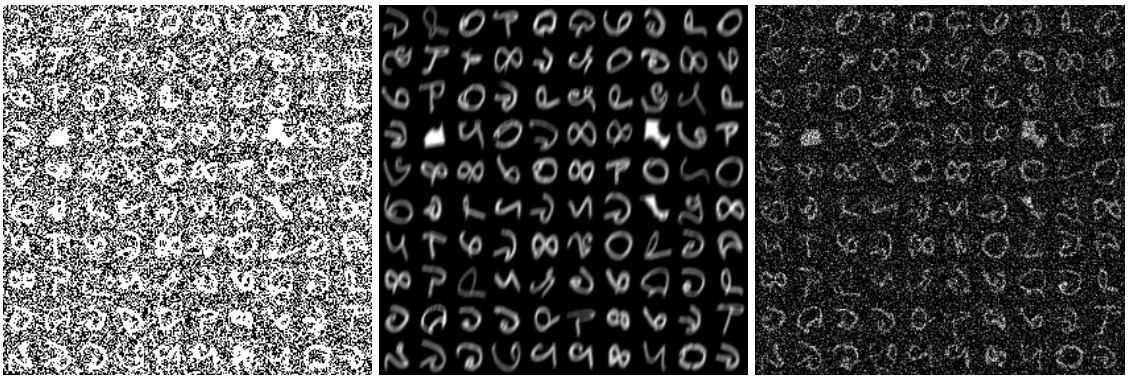
The numerals in the Bangla dataset are first thresholded using a local adaptive mean filter. The thresholded images are then complemented and we extract the largest connected component. Then we find the center of mass of the largest connected component and the corresponding bounding box of the numeral and then use this information to center the image. The centered image is then padded with 10 pixels on all sides. Finally, the images are resized to 28×28 pixels.

- **Data Augmentation**

Following the procedure defined in [16], we create a synthetic dataset by using rotation and blurring on the original Bangla dataset. For the rotation transformation, each sample is randomly rotated by an angle which lies in the range 5° and 10° and another in the range -10° and -5°. All the original and rotated training samples generated above are blurred by applying a Gaussian blurring kernel with a mean value of $\mu = 0.75$ and a standard deviation value of $\sigma = 0.33$. So, the original images along with the rotated and blurred images form the final training dataset. These images are in turn added with noise and form our noisy Bangla dataset.

4.3 Probabilistic Quadtrees for Learning Sparse Representations

A quadtree is a data structure which takes the form of a tree in which all the internal nodes have four child nodes. Quadtrees were first proposed in [34] as a technique for storing key-



(a) Bangla with Additive White Gaussian Noise

(b) Bangla with Motion Blur

(c) Bangla with AWGN and reduced contrast

Figure 4.2: Example images from the noisy Bangla dataset created as part of the experiments.

value pairs indexed with composite keys. We propose a novel technique based on probabilistic quadtrees that can perform dimensionality reduction on a dataset in a probabilistically sound way. We learn the structure of the quadtree from the samples of a dataset. A quadtree splits each image into four equi-sized windows, and then performs a test of homogeneity on each image window. If a block meets the homogeneity criterion, it is not divided into sub-windows. If otherwise, it fails to meet the criterion, it is again divided into four sub-windows, and the test criterion is in turn applied to those smaller windows. This process is repeated on all the sub-windows until each meets the homogeneity criterion. The resulting data structure can have windows of several different sizes. The homogeneity criterion can be defined as follows - Split a block if the difference between the highest value of the elements of the block and the least value exceeds a threshold τ. Threshold τ is chosen to be a value lying between 0 and 1 (chosen here as 0.27 by experiments). Denoting the homogeneity criterion for sample d as H_d, this can be formally presented as follows:

$$H_d = \begin{cases} true, & \text{if} \max_{i \in d}(i) - \min_{i \in d}(i) \leq \tau \mid \tau \in [0,1] \\ false, & \text{if} \max_{i \in d}(i) - \min_{i \in d}(i) > \tau \mid \tau \in [0,1] \end{cases} \tag{4.1}$$

Alternatively, the homogeneity criterion can be considered proportional to the standard deviation of the probability distribution of the dataset. So, higher the standard deviation, higher the

average texture of a block and higher is the probability of the block being divided into sub-blocks.

In the learned quadtree structure for a given dataset, a node is divided into smaller windows if the homogeneity criterion is not met for any sample in the dataset. The node is not divided into smaller windows only if the homogeneity criterion is met by all samples in the dataset.

We can consider each node of the quadtree as a binary random variable X, which can take one of two values 1 or 0 based on whether it is divided into smaller windows or not. So, for a total of N samples in dataset D, the random variable X may assume one of $N + 1$ possible split states: one value for each of the samples not meeting the homogeneity criterion, and one value indicating that all samples meet the homogeneity criterion. This can be formally presented as follows:

$$X = \begin{cases} 1, & \text{if } \exists d \in D \mid D = \{d_0, d_1, d_2, ..., d_N\} \cap \{H_d = false\} \\ 0, & \text{if } \forall d \in D \mid D = \{d_0, d_1, d_2, ..., d_N\} \cap \{H_d = true\} \end{cases} \tag{4.2}$$

Once learned, the probabilistic quadtree helps in performing dimensionality reduction of the data, which captures the statistics of the training samples in the dataset. A depth first search on the learned tree yields a linear vector that is then fed into an unsupervised learning framework.

4.4 Deep Belief Network for Feature Learning

Deep Belief Network (DBN) consists of multiple layers of stochastic, latent variables trained using an unsupervised learning algorithm followed by a supervised learning phase using Feed-forward Backpropagation Neural Networks. In the unsupervised pre-training stage, each layer is trained using a *Restricted Boltzmann Machine* (RBM). Once trained, the weights and biases of the Deep Belief Net are used to initialize the parameters of a Deep Neural Network [13]. A Neural Network which has been initialized in this manner converges at a much faster rate to the optimal solution as compared to an uninitialized one. A DBN is a graphical model [59] where neurons of the hidden layer are conditionally independent of each other given a particular configuration of the visible layer and vice versa. A DBN can be trained layer-wise by iteratively maximizing the conditional probability of the input vectors or visible vectors given the hidden

vectors and a particular set of layer weights. As shown in [48], this layer-wise training can help in improving the variational lower bound on the probability of the input training data, which in turn leads to an improvement of the overall generative model. We first provide a formal introduction to the Restricted Boltzmann Machine. The RBM can be denoted by the energy function:

$$E(u, v) = -\sum_{i} a_i u_i - \sum_{j} b_j v_j - \sum_{i} \sum_{j} v_j w_{i,j} u_i \tag{4.3}$$

where, the RBM consists of a matrix of layer weights $W = (w_{i,j})$ between the hidden units v_j and the visible units u_i. The a_i and b_j are the bias weights for the visible units and the hidden units respectively. The RBM takes the structure of a bipartite graph and hence it only has inter-layer connections between the hidden or visible layer neurons but no intra-layer connections within the hidden or visible layers. So, the visible unit activations are mutually independent given a particular set of hidden unit activations and vice versa [23]. Hence, by setting either v or u constant, we can compute the conditional distribution of the other as follows:

$$P(v_j = 1|u) = \sigma(\sum_{i=1}^{p} w_{i,j} u_i + b_j) \tag{4.4}$$

$$P(u_i = 1|v) = \sigma(\sum_{j=1}^{q} w_{i,j} v_j + a_i) \tag{4.5}$$

where, σ denotes the log sigmoid function:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{4.6}$$

The training algorithm maximizes the expected log probability assigned to the training dataset D. So if the training dataset D consists of the visible vectors u, then the objective function is as follows:

$$\underset{W}{\operatorname{argmax}}\, E\Big[\sum_{u \in U} \log P(u)\Big] \tag{4.7}$$

A Restricted Boltzmann Machine is trained using a *Contrastive Divergence* algorithm [23]. Once trained the DBN is used to initialize the weights of a feedforward backpropagation neural network that is then used for classification. The neural network gives an estimate of the posterior probabilities of the class labels, given the input vectors. As illustrated in [18], the output of a neural network trained by minimizing the sum of squares error function approximates the conditional averages of the target data

$$y_k(x) = \langle t_k|x\rangle = \int t_k p(t_k|x)dt_k \tag{4.8}$$

Here, t_k are the set of target values that represent the class membership of the input vector x_k. For a binary classification problem, in order to map the outputs of the neural network to the posterior probabilities of the labeling, we use a single output y and a target coding that sets $t^n = 1$ if x^n is from class C_1 and $t^n = 0$ if x^n is from class C_2. The target distribution would then take the form

$$p(t_k|x) = \delta(t-1)P(C_1|x) + \delta(t)P(C_2|x) \tag{4.9}$$

Here, δ represents the Dirac delta function that satisfies the following conditions [**?**] $\delta(x) = 0$ if $x \neq 0$ and

$$\int_{-\infty}^{\infty} \delta(x)\,\mathrm{d}x = 1 \tag{4.10}$$

From 4.8 and 4.9, we get

$$y(x) = P(C_1|x) \tag{4.11}$$

So, the network output $y(x)$ represents the posterior probability of the input vector x having the class membership C_1 and the probability of the class membership C_2 is given by $P(C_2|x) = 1 - y(x)$. This argument can easily be extended to multiple class labels for a generalized multi-

class classification problem like MNIST.

The dimensionality reduction using probabilistic quadtrees helps improve the discriminative power of the DBN based classifier significantly.

- **Gibbs' Sampling in RBM**

Gibbs' Sampling is a Markov Chain Monte Carlo method that is used to sample from a distribution when it is difficult to do direct sampling. Once sampled, these observations can be used to estimate the joint probability distribution or the marginal probability of a variable. In the context of an RBM, Gibbs' sampling is useful to estimate the log-likelihood gradient of the data. Generating samples from the model is equivalent to sampling from the DBN since the DBN is formed by stacking multiple layers of RBMs.

Gibbs' sampling in unrestricted Boltzmann Machines are computationally intensive because we need to perform sampling both for the input neurons and the joint distributions of the input and hidden neurons. Also for unrestricted Boltzmann machines, the number of Gibbs' sampler steps is equal to the number of units in the neural network. On the other hand, in an RBM, the hidden layer units are conditionally independent for a given configuration of the visible layer units and vice versa. So, the neurons can be updated in parallel and hence a Gibbs sampling in an RBM consists of only two sub-steps - 1) Sample hidden vectors given the visible vectors. Also, in RBM, the free energy state of the input neurons doesn't need to be sampled but can be directly calculated analytically.

- **Contrastive Divergence**

The contrstive divergence algorithm approximates the the log-likelihood gradient of the data. This has been shown to be very useful for the training of Restricted Boltzmann machines. To obtain this approximation, the average over all possible samples is replaced with a single sample. With frequent updates in parameter values (after seeing one or a few training samples), there is automatic averaging going on which and this partially nullifies the effect of the increased variance introduced into the model due to the use of one or a few MCMC samples instead of using all the samples from the chain. However, this cancellation is only partial and hence there

is still some extra variance introduced during the approximation. Running the full MCMC chain is computationally infeasible and hence we resort to k-step Contastive Divergence (CD-k). The CD-k algorithm performs another approximation. It runs the MCMC chain for only k steps. This reduces the computational cost greatly however leads to a higher bias in the model. The CD-k update rule can be noted as follows:

$$\triangle\theta = \frac{\partial E(v)}{\delta\theta} + \frac{\partial E(\tilde{v})}{\delta\theta} \tag{4.12}$$

where, E is the free energy and v is the observed sample and $\tilde{v}$ is the sample obtained after k-steps of MCMC. The bias tends to 0 as $k \to \infty$, since at $k = \infty$ the model distribution converges to the equilibrium distribution.

Another interesting observation about the CD-k algorithm is that when the distribution of the model approaches the equilibrium distribution, i.e., when $P \approx \tilde{P}$, then when we start the MCMC chain from a sample x, belonging to $\tilde{P}$ then the model is already good at approximating the underlying distribution and we need only one step to generate a sample of the unbiased estimator with underlying distribution P.

It has been shown that in the CD-k algorithm even $k = 1$ leads to good results. This has been shown both theoretically and empirically. The theoretical results are supported by the argument that the contrastive divergence algorithm approximates the first k terms of a series which provides an approximation of the log-likelihood gradient of the data. The contrastive divergence algorithm can be seen as approximating the gradient of the log-likelihood of the data centered around the training sample x_1. As we increase k, the underlying distribution of the reconstructed sample x_{k+1} moves further from X and closer towards the model distribution.

4.5 Results and Comparative Studies

Various network architectures along with the test set error for the traditional DBN framework and the probabilistic quadtree based framework on the MNIST and the three n-MNIST datasets are listed in Tables 4.1 and 4.2. From the Tables, it is evident that our best performing

	MNIST		n-MNIST with AWGN	
Architecture (Neurons)	**Test Error DBN(%)**	**Test Error Ours(%)**	**Test Error DBN(%)**	**Test Error Ours(%)**
50-50	4.64	2.93	89.95	13.41
100-100	3.01	2.45	91.43	12.01
150-150	2.34	2.21	89.95	13.49
200-200	2.08	1.96	88.49	10.56
250-250	1.93	1.83	88.49	13.00
300-300	2.02	1.80	68.18	11.24
350-350	1.96	1.74	90.31	13.15
400-400	1.95	1.67	49.27	10.96
450-450	1.93	**1.38**	**32.26**	12.62
500-500	**1.86**	1.43	69.68	**9.93**

Table 4.1: Test Error of a traditional DBN and our framework with various architectures on MNIST and n-MNIST with AWGN

	n-MNIST with Motion Blur		n-MNIST with AWGN and Reduced Contrast	
Architecture (Neurons)	**Test Error DBN(%)**	**Test Error Ours(%)**	**Test Error DBN(%)**	**Test Error Ours(%)**
50-50	5.64	4.17	10.21	9.29
100-100	4.68	3.31	**9.43**	9.21
150-150	3.99	3.29	16.40	9.00
200-200	3.74	3.03	15.57	8.79
250-250	3.74	**2.60**	52.31	8.94
300-300	**3.50**	3.04	32.29	8.28
350-350	3.82	2.91	86.31	8.90
400-400	3.74	3.01	68.78	8.31
450-450	3.91	2.75	51.32	8.36
500-500	3.66	2.83	68.19	**7.84**

Table 4.2: Test Error of a traditional DBN and our framework with various architectures on n-MNIST with Motion Blur; and with AWGN and Reduced Contrast

network outperforms the best traditional Deep Belief Network on both the MNIST and n-MNIST datasets. On the MNIST dataset, our best network exhibits a relative improvement of $\sim$25% over the traditional DBN. For the n-MNIST dataset, it provides a relative improvement of $\sim$36% for Additive White Gaussian Noise (AWGN), $\sim$26% for Motion Blur and $\sim$12% for AWGN and Reduced contrast. Table 4.3 and Table 4.4 show the test error rates of the traditional DBN based framework and the quadtree based DBN framework on the noisy Bangla datasets. As seen in the

	Noisy Bangla with AWGN	
Architecture (Neurons)	**Test Error DBN(%)**	**Test Error Ours(%)**
50-50	12.29	11.94
100-100	10.86	9.6
150-150	10.54	10.43
200-200	9.97	9.62
250-250	9.94	9.88
300-300	9.4	9.13
350-350	9.97	9.77
400-400	**8.91**	**8.66**
450-450	9.54	9.51
500-500	9.2	9.02

Table 4.3: Test Error of a traditional DBN and our framework with various architectures on the noisy Bangla dataset with AWGN

	Noisy Bangla with Motion Blur		Noisy Bangla with AWGN and Reduced Contrast	
Architecture (Neurons)	**Test Error DBN(%)**	**Test Error Ours(%)**	**Test Error DBN(%)**	**Test Error Ours(%)**
50-50	10.6	10.46	22.46	14.97
100-100	9.31	8.83	18.34	**12.69**
150-150	8.54	8.26	17.37	13.69
200-200	8.4	7.91	17.06	13.08
250-250	8.34	7.71	17.26	13
300-300	8.13	8.11	17.14	13.47
350-350	8	7.8	**16.92**	13.2
400-400	8.2	7.63	17.3	13.8
450-450	**7.91**	7.75	17.21	13.73
500-500	8.28	**7.34**	17.38	13.95

Table 4.4: Test Error of a traditional DBN and our framework with various architectures on noisy Bangla dataset with Motion Blur; and with AWGN and Reduced Contrast

Dataset	**Mean Normalized Contrast**
MNIST	1.0
Offline Bangla	0.87

Table 4.5: Mean contrast of the MNIST and Bangla numeral datasets.

tables, the best quadtree based DBN framework provides a relative improvement of $\sim$3% over the best traditional DBN architecture for the noisy Bangla dataset with AWGN. For the motion blur dataset, it produces a relative improvement of $\sim$8% while for the noisy Bangla dataset with

AWGN and reduced contrast, it produces a relative improvement of $\sim$33%. It is interesting to observe that the relative improvement in performance using the quadtree based framework is highest for n-MNIST with AWGN and lowest for n-MNIST with AWGN and reduced contrast. On the contrary, for the noisy Bangla dataset, the relative improvement is the highest for the dataset with AWGN and reduced contrast and lowest for the AWGN dataset. This is because the Bangla dataset generated by the pre-processing and data augmentation stages has a lower contrast than the MNIST dataset as seen in Table 4.5 and hence the application of reduced contrast on top of the already low contrast dataset creates a noisy dataset which is much more difficult to be handled by the traditional DBN. Hence, the quadtree based DBN produces significant improvement over the traditional DBN for the noisy Bangla dataset with AWGN and reduced contrast. Due to this same reason, the mean test error rates of the various architectures of both the traditional DBN and the quadtree based DBN on the reduced contrast-AWGN dataset are higher than the AWGN dataset and the motion blur dataset.

Chapter 5
A Theoretical Analysis of Deep Neural Networks for Texture Classification[1]

In this chapter, we investigate the use of Deep Neural Networks for the classification of image datasets where texture features are important for generating class-conditional discriminative representations. To this end, we first derive the size of the feature space for some standard textural features extracted from the input dataset and then use the theory of Vapnik-Chervonenkis dimension to show that hand-crafted feature extraction creates low-dimensional representations which help in reducing the overall excess error rate. As a corollary to this analysis, we derive for the first time upper bounds on the VC dimension of Convolutional Neural Network as well as Dropout and Dropconnect networks and the relation between excess error rate of Dropout and Dropconnect networks. The concept of *intrinsic dimension* is used to validate the intuition that texture-based datasets are inherently higher dimensional as compared to handwritten digits or other object recognition datasets and hence more difficult to be shattered by neural networks. We then derive the mean distance from the centroid to the nearest and farthest sampling points in an n-dimensional manifold and show that the *Relative Contrast* of the sample data vanishes as dimensionality of the underlying vector space tends to infinity.

5.1 Introduction

Texture is a key recipe for various object recognition tasks which involve texture-based imagery data like Brodatz [101], VisTex [114], Drexel [80], KTH [103], UIUCTex [63] as well as forest species datasets [81]. Texture characterization has also been shown to be useful in addressing other object categorization problems like the Brazilian Forensic Letter Database (BFL) [22] which was later converted into a *textural representation* in [45]. In [30], a similar approach was used to find a textural representation of the Latin Music Dataset [90].

Over the last decade, Deep Neural Networks have become increasingly popular due to their

[1]Adapted from the paper titled "A Theoretical Analysis of Deep Neural Networks for Texture Classification" accepted for publication at the International Joint Conference on Neural Networks, IJCNN 2016 [11].

ability to learn data representations in both supervised and unsupervised settings and generalize to unseen data samples using hierarchical representations. A notable contribution in *Deep Learning* is a *Deep Belief Network*(DBN) formed by stacking together *Restricted Boltzmann Machines* [48]. Another closely related approach, which has gained much traction over the last decade, is the Convolutional Neural Network (CNN) [67]. CNN's have been shown to outperform DBN in classical object recognition tasks like MNIST [105] and CIFAR [61]. Despite these advances in the field of Deep Learning, there has been limited success in learning textural features using Deep Neural Networks. Does this mean that there is some inherent limitation in existing Neural Network architectures and learning algorithms?

In this chapter, we try to answer this question by investigating the use of Deep Neural Networks for the classification of texture datasets. First, we derive the size of the feature space for some standard textural features extracted from the input dataset. We then use the theory of Vapnik-Chervonenkis (VC) dimension to show that hand-crafted feature extraction creates low-dimensional representations, which help in reducing the overall excess error rate. As a corollary to this analysis we derive for the first time upper bounds on the VC dimension of Convolutional Neural Network as well as Dropout and Dropconnect networks and the relation between excess error rate of Dropout and Dropconnect networks. The concept of *intrinsic dimension* is used to validate the intuition that texture-based datasets lie on an inherently higher dimensional manifold as compared to handwritten digits or other object recognition datasets and hence more difficult to be classified/shattered by neural networks. To highlight issues associated with the Curse of Dimensionality of texture datasets, we provide theoretical results on the mean distance from the centroid to the nearest and farthest sampling points in n-dimensional manifolds and show that the *Relative Contrast* of the sample data vanishes as dimensionality of the underlying vector space tends to infinity. Our theoretical results and empirical analysis show that in order to classify texture datasets using Deep Neural Networks, we need to either integrate them with handcrafted features or devise novel neural architectures that can learn features from the input dataset that resemble these handcrafted texture features.

5.2 VC dimension of Deep Neural Networks and Classification Accuracy

VC dimension was first proposed in [97] and was later applied to Neural Networks in [6]. It was noted in [17] that the VC dimension proposed for Neural Networks is also applicable to Deep Neural Networks. It was shown in [6] that for neural nets with sigmoidal activation function, the VC-dimension is loosely upper-bounded by $O(w^4)$ where w is the number of free network parameters. Given a classification model M, the VC-dimension of M is equivalent to the highest number of points in a sample space that can be divided by a decision boundary using M. This phenomenon of spliting or dividing points based on decision boundaries is called shattering.

We estimate the size of the sample space composed of the various features extracted from the textural Co-occurrence Matrices (Haralick features) following those proposed in [46]. We then use the theory of VC dimension to show that texture feature extraction creates low dimensional representations which help in reducing the overall excess error rate.

5.2.1 Sample complexity of Haralick features and the fat-shattering dimension[2]

For simplicity, we consider intensity image with a single channel and Gray-Level Co-occurrence Matrix (GLCM) which can be extended to multi-channel images and general Color Co-occurrence Matrices (CCM) without loss of generality. For $n \times n$ images with k color levels, the following results can be derived[3].

Proposition 5.2.1. *If $x_1, x_2, \ldots x_{k^2}$ be the values of the $k \times k$ GLCM matrices, then the number of distinct matrices is given by $\binom{n^2+k^2-1}{k^2-1}$.* □

Proof. The number of distinct GLCM matrices is the same as the number of non commutative ways to write n^2 as the sum of k^2 non-negative integers. Assume a line of $n^2 - k^2 + 1$ positions where each position can contain a ball or a divider. If we have n^2 (identical) balls and $k^2 - 1$

[2]For a detailed description of the various GLCM metrics defined in this section and the notations used, we refer the reader to [46]

[3]A proof of these results follows from simple counting arguments.

dividers we can split the balls into k^2 groups by choosing positions from the dividers: $\binom{n^2+k^2-1}{k^2-1}$. The size of each group corresponds to one of the non-negative integers in the sum. □

Proposition 5.2.2. *The number of distinct values for GLCM angular 2^{nd} moment is* $n^4 - \left(\left\lfloor \frac{n^2}{k^2} \right\rfloor^2 \times \left(k^2 - 1\right) + \left(n^2 - \left(k^2 - 1\right)\left\lfloor \frac{n^2}{k^2} \right\rfloor\right)^2 + 1\right)$ □

Proof. GLCM angular 2^{nd} moment is given by $\sum_i \sum_j p(i,j)^2$. Now, the angular 2^{nd} moment can assume only integral values. Maxima occurs when one pixel has the value n^2 and rest are 0. Minima occurs when n^2 is divided between $k^2 - 1$ points and the rest is at the one remaining point. For n^2 divided into $k^2 - 1$ points, Number of distinct values is given by $\left(\left\lfloor \frac{n^2}{k^2} \right\rfloor\right)^2 (k^2 - 1)$ and for the remaining pixel, we have number of distinct values as $(n^2 - \left\lfloor \frac{n^2}{k^2} \right\rfloor^2 (k^2 - 1))^2$. Adding these two and subtracting it from n^4, we get the final result as $n^4 - \left(\left\lfloor \frac{n^2}{k^2} \right\rfloor^2 \times \left(k^2 - 1\right) + \left(n^2 - \left(k^2 - 1\right)\left\lfloor \frac{n^2}{k^2} \right\rfloor\right)^2 + 1\right)$. □

Proposition 5.2.3. *The number of distinct values of GLCM correlation is* $n^2k^2 - n^2 - \frac{k^2}{2} + \frac{k}{2} + 1$. □

Proof. GLCM correlation is given by $\frac{\sum_i \sum_j ijp(i,j) - \mu_x \mu_y}{\sigma_x \sigma_y}$. Minima is n^2 when $p(1,1)$ is n^2 and maxima is k^2n^2 when $p(k,k)$ is n^2. The range would then be $k^2n^2 - n^2 + 1$. For other cases, when, $P(k,k)$ is $n^2 - 1$ and $P(k,k-1)$ is 1, Maxima is k^2n^2 and Minima is $k^2n^2 - k$. Similarly, by induction we can show that in the general case, we exclude $\frac{k(k-1)}{2}$ terms. Therefore, a tight upper bound on the solution would be $k^2n^2 - n^2 + 1 - \frac{k(k-1)}{2}$ distinct GLCM correlation values. □

Proposition 5.2.4. *The number of distinct values of GLCM sum average is* $2n^2k - 2n^2 + 1$. □

Proof. GLCM sum average is given by $\sum_{i=2}^{2k} ip_{x+y}(i)$ where $p_{x+y}(t) = \sum_{\substack{i=1 \\ i+j=t}}^{k} \sum_{j=1}^{k} p(i,j)$. Maxima is $2n^2k$ which occurs when p(i,j) attains the maximum value, i.e., n^2 at $i = j = k$. Minima is $2n^2$ which occurs when p(i,j) attains the maximum value i.e., n^2 at $i = j = 1$. Therefore, the number of distinct values of GLCM sum average is $2n^2k - 2n^2 + 1$. □

Proposition 5.2.5. *The number of distinct values of GLCM contrast is* $n^2k^2 + n^2 - 2n^2k + 1$. □

Proof. GLCM contrast is given by $\sum_{n=0}^{k-1} n^2 \sum_{\substack{i=1 \\ |i-j|=n}}^{k} \sum_{j=1}^{k} p(i,j)$. The maxima occurs when the entire sum n^2 of the GLCM occurs at one pixel at the top right or at one pixel at the bottom left. These are the two points in the GLCM where $|i-j| = k-1$. So, Maxima is $(k-1)^2 n^2 = k^2 n^2 + n^2 - 2n^2 k$. Similarly, Minima occurs when the entire GLCM sum n^2 is distributed among points only along the diagonal where $|i-j| = 0$ hence resulting in a minima of 0. Therefore, distinct number of GLCM contrast values is $n^2 k^2 + n^2 - 2n^2 k + 1$. □

From proposition 5.2.2 through 5.2.5, it can be seen that in the general case, number of distinct Haralick features is given by $O(n^2 k^2 + n^4)$. For deep neural networks, the VC dimension is upper bounded by $O(w^4)$ according to [6]. Now, we can pick the number of adjustable parameters w to be such that $n \leq k \leq w$ or $k \leq n \leq w$. In both cases, we have $O(n^2 k^2) \leq O(w^4)$ and $O(n^4) \leq O(w^4)$ which gives $O(n^2 k^2 + n^4) \leq O(w^4)$. Hence, the number of possible distinct values for the GLCM based feature vectors is much lower than the VC dimension of such a network. So, we can effectively argue that the VC-dimension of a Deep Neural Network with w adjustable parameters is such that it can shatter the metrics formed using GLCM - the only prerequisite being that we select a network with the number of adjustable parameters as an upper bound for the input data dimensionality and the number of distinct gray levels in the color channel. On the other hand, in order to shatter the raw image vectors, the effective VC dimension of the network should be at least of the order of $O(k^{n^2})$. So, for the GLCM based features, we need Neural Networks with smaller VC dimension as compared to raw vectors. Also, in the next section, we show that with increase in VC dimension of the network, the excess error rate increases. So, the composite learning model formed by the integration of GLCM based features and Deep Neural Networks have lower excess error rate as compared to Deep Neural Networks combined with raw image pixels.

5.3 Input Data Dimensionality and bounds on the test error

In this section, we derive the relation between input data dimensionality and upper bound Γ on the excess error rate of the Deep Neural Network. As a corollary to this analysis we derive

for the first time upper bounds on the VC dimension of Convolutional Neural Network as well as Dropout and Dropconnect networks and show that the upper bound Γ on the excess error rate of the Dropout networks is lower than that of DropConnect.

Lemma 5.3.1. *With increase in the input data dimensionality, the dimensionality of the optimal model increases.*

Proof. As shown in [75], for input data dimensionality d and model dimensionality p, the number of cells formed by p planes in d space is given by

$$C(p,d) = \sum_{i=0}^{min(p,d)} \binom{p}{i} = \begin{cases} 2^p & , p \leq d \\ \sum_{i=0}^{d} \binom{p}{i} & , p > d. \end{cases} \tag{5.1}$$

Now, the number of cells per dimension gives the number of divisions of the model space along each dimension and can be approximated as $C(p,d)^{1/d}$. This in turn is equal to the number of class labels c. Therefore, for a given classification problem with c class labels, we have, $C(p,d)^{1/d} = c$ and hence, we have

$$c = C(p,d)^{1/d} = \begin{cases} 2^{p/d} & , p \leq d \\ (\sum_{i=0}^{d} \binom{p}{i})^{1/d} & , p > d. \end{cases} \tag{5.2}$$

From equation 5.2, it follows that with increase in data dimensionality d, the model dimensionality p should increase, given a fixed classification problem with c class labels. □

Lemma 5.3.2. *With increase in the dimensionality of the model, its VC dimension increases.*

Proof. This statement follows from the VC dimension bounds of both a Deep Neural Network and a Deep Convolutional Neural Network (CNN). The upper bound for the VC dimension of a Deep Neural Network is given by $O(w^4)$ [6] and the upper bound on the VC dimension of a Convolutional Neural Network is given by $O\left(\frac{m^4 k^4 s^{2l-2}}{l^2}\right)$. The result for the Deep Neural Network follows from [6], where it is noted that the VC dimension of Deep Neural Networks with

sigmoidal activation functions is given by $O(t^2d^2)$ which reduces to $O(w^4)$. The result of the VC bound for the CNN along with the proof is detailed in Theorem 5.3.3 below. □

Theorem 5.3.3. *The VC dimension of a Convolutional Neural Network is upper bounded by* $O\left(\frac{m^4k^4s^{2l-2}}{l^2}\right)$.

Proof. From Theorem 5 and Theorem 8 in [6], it can be seen that for the parameterized class $F = \{x \longmapsto f(\theta, x) : \theta \in \mathbb{R}^d\}$ with the arithmetic operations $+, -, \times, /$ and the exponential operation $\alpha \longmapsto e^\alpha$ on the space of real numbers, conditional operators $=, \neq, >, <, \geq$, and $\leq$ on the space of real numbers and an output of 0/1, VCDim(F) = $O(t^2d^2)$. Here, t is the number of operations and d is the dimensionality of the adjustable parameter space. Now, for the CNN, input size is n, kernel size is k, sampling factor is s and we assume convolution kernel step size as 1 for simplicity. So, we have $\frac{\frac{\frac{n-k}{s}-k}{s}}{s} \ldots$ upto l layers which in turn is equal to 1 for a binary classification problem[4]. Now, in the simplest case, we have a CNN with one convolutional layer followed by one subsampling layer (c-s). Hence, $\frac{n-k}{s} = 1 \implies n = s + k$. For a CNN with the configuration (c-s-c-s), we have,

$$\frac{\frac{n-k}{s} - k}{s} = 1 \implies n = k + s(s + k) = s^2 + ks + k \tag{5.3}$$

Continuing this pattern, we have in the general case,

$$\begin{aligned} \frac{\frac{\frac{n-k}{s}-k}{s} - k}{s} \ldots \text{upto } l \text{ layers} = 1 \\ \implies n = s^l + ks^{l-1} + ks^{l-2} + \ldots + ks + k \end{aligned} \tag{5.4}$$

Now, let $m_1, m_2, \ldots m_l$ be the number of maps in the various layers of a CNN and $t = t_1 + t_2 + \ldots + t_l$ be the total number of operations. Now, for layer 1, number of operations $t_1 = m_1(n-k)$, for layer 2, number of operations $t_2 = m_2(\frac{n-k}{s} - k)$, and so on. Therefore, Total number of

[4]Note that for simplifying the algebra, we consider only the convolutional and subsampling layers of a CNN. This analysis can be extended to hybrid architectures with other types of layers (e.g., fully connected) by adjusting t and d.

operations

$$t = m_1(n-k) + \ldots + m_l(\frac{\frac{n-k}{s} - k}{s} \ldots \text{to } l \text{ layers})$$
$$= m_1(ks + ks^2 + \ldots + ks^{l-1} + s^l) + m_2(ks + ks^2 + \ldots$$
$$+ ks^{l-2} + s^{l-1}) + \ldots + m_l s \quad (5.5)$$

Also, dimensionality of parameter space is given by $d = m_1k + m_2k + \ldots + m_lk$. Now, for simplifying, if we assume that the number of maps in the layers $m_1 = m_2 = \ldots = m_l = \frac{m}{l}$, then, we have

$$t = \frac{m}{l}(n-k) + \frac{m}{l}(\frac{n-k}{s} - k) + \ldots + \frac{m}{l}(\frac{\frac{\frac{n-k}{s}-k}{s} - k}{s}$$
$$\ldots \text{upto } l \text{ layers}) = \frac{mks^2(s^{l-1}-1)}{l(s-1)^2} + \frac{ms(s^l-1)}{l(s-1)}$$
$$= O(\frac{mks^{l-1}}{l}) \quad (5.6)$$

$$\text{Also, } d = O(mk) \quad (5.7)$$

From equation 5.6 and equation 5.7, we have VCdim$_{CNN}$ = $O\left(\frac{m^4k^4s^{2l-2}}{l^2}\right)$

□

Theorem 5.3.4. *Upper bound on excess error rate $\mathcal{E}$ increases with increase in VC dimension given fixed number of training sample points N.*

Proof. Using the theory of VC dimension, we have Excess error rate

$$\mathcal{E} \leq \sqrt{\frac{d(\log(2T/d) + 1) - \log(\theta/4)}{T}} \quad (5.8)$$

where, d is the VC dimension of the learning model, T is the number of training sample points and $0 \leq \theta \leq 1$. From equation 5.8, the result follows. □

Theorem 5.3.5. *For a given Dropout network with probability of dropout p and number of adjustable paramaters in the network being w, the VC dimension of the network is upper bounded by $O\Big((1-p)^8 w^4\Big)$.*

Proof. For a neural network having number of neurons $n = n_1 + n_2 + n_3 + \ldots + n_l$, the number of adjustable paramaters w is given by

$$w = n_1 n_2 + n_2 n_3 + n_3 n_4 + \ldots + n_{l-1} n_l \tag{5.9}$$

For a given dropout fraction p, each neuron in the network can be dropped by a probability of p. So the effective number of neurons in the Dropout network

$$\widetilde{n} = (1-p)(n_1 + n_2 + \ldots + n_l) \tag{5.10}$$

Now, we can split the effective number of neurons in each layer as $\widetilde{n_1} = (1-p)n_1,\ \ldots,\ \widetilde{n_l} = (1-p)n_l$.

Therefore, $\widetilde{w} = \widetilde{n_1}\widetilde{n_2} + \widetilde{n_2}\widetilde{n_3} + \ldots + \widetilde{n_{l-1}}\widetilde{n_l}$

$$= (1-p)^2(n_1 n_2 + \ldots + n_{l-1} n_l) = (1-p)^2 w \tag{5.11}$$

Now, given that $\text{VCDim}_{Dropout} = O\Big(\widetilde{w}^4\Big)$ we have, $\widetilde{w}^4 = ((1-p)^2)^4 w^4 = O\Big((1-p)^8 w^4\Big)$. So,

$$\text{VCDim}_{Dropout} = O\Big((1-p)^8 w^4\Big) \tag{5.12}$$

□

Theorem 5.3.6. *For a given Dropconnect network with probability of drop p and number of adjustable paramaters in the network being w, the VC dimension of the network is upper bounded by $O\Big((1-p)^4 w^4\Big)$.*

Proof. Since in a Dropconnect network, each weight can be dropped by a probability of p, so,

effective number of adjustable parameters in the Dropconnect network is given by $\widetilde{w} = (1-p)w$. Now, given that, $\widetilde{w}^4 = (1-p)^4 w^4 = O\Big((1-p)^4 w^4\Big)$, we have,

$$\text{VCDim}_{Dropconnect} = O\Big((1-p)^4 w^4\Big) \tag{5.13}$$

□

Theorem 5.3.7. *For a given drop probability p, the number of adjustable paramaters in the network being w, the excess error rate being $\mathcal{E}$ and the upper bounds on the error rates of the Dropout and Dropconnect networks being $\Gamma_{Dropout}$ and $\Gamma_{Dropconnect}$ respectively, we have $\Gamma_{Dropout} \leq \Gamma_{Dropconnect}$.*

Proof. From Equation 5.8 and 5.12, we have

$$\text{Excess error rate}, \mathcal{E} \leq \sqrt{\frac{d(\log(2T/d)+1) - \log(\theta/4)}{T}} \tag{5.14}$$

Therefore, upper bound on the error rate $\Gamma_{Dropout}$

$$= \sqrt{\frac{1}{T}(1-p)^8 w^4 \Big[\log\Big(\frac{2T}{(1-p)^8 w^4}+1\Big)\Big] - \log(\theta/4)} \tag{5.15}$$

Similarly, from Equation 5.8 and 5.13, we have for the Dropconnect network, $\Gamma_{Dropconnect}$

$$= \sqrt{\frac{1}{T}(1-p)^4 w^4 \Big[\log\Big(\frac{2T}{(1-p)^4 w^4}+1\Big)\Big] - \log(\theta/4)} \tag{5.16}$$

For a given w, T and probability of drop p with $0 \leq p < 1$, it can be easily shown that the upper bounds on the excess error rates of the Dropout and Dropconnect networks are related as

$$\Gamma_{Dropout} \leq \Gamma_{Dropconnect} \tag{5.17}$$

□

5.4 What is the Difference between Object Recognition Datasets and Texture-based Datasets in terms of Dimensionality?

We argue that object recognition datasets lie on a much lower dimensional manifold than texture datasets. Hence, even if Deep Neural Networks can effectively shatter the raw feature space of object recognition datasets, the dimensionality of texture datasets is such that without explicit texture-feature extraction, these networks cannot shatter them. In order to estimate the dimensionality of the datasets, we use the concept of *intrinsic dimension*[71].

5.4.1 Intrinsic Dimension Estimation using the Maximum Likelihood algorithm

The *intrinsic dimension* of a dataset represents the minimum number of variables that are required to represent the data. We use the Maximum Likelihood algorithm proposed in [71] to estimate the Intrinsic dimension of various datasets. The results for the various datasets and the Haralick features extracted are listed in Table 5.1 and 5.2. The DET dataset [86] is a subset of the Imagenet dataset.

Dataset	MNIST	CIFAR10	DET
Intrinsic Dim.	9.96	15.9	17.01

Table 5.1: Intrinsic Dimension estimation using MLE on the MNIST, CIFAR-10 and DET datasets

Dataset	**Brodatz**	**VisTex**	**KTH**
Intrinsic Dimension (Raw Vectors)	34.87	44.81	43.69
Intrinsic Dimension (Texture Features)	4.03	3.84	3.73
Dataset	**KTH2**	**Drexel**	**UIUCTex**
Intrinsic Dimension (Raw Vectors.)	54.19	30.26	33.64
Intrinsic Dimension (Texture Features)	3.93	4.24	4.57

Table 5.2: Intrinsic Dimension estimation using MLE on the 6 texture datasets

From Table 5.1 and 5.2, we can see that the intrinsic dimensionality of the texture datasets (Brodatz, VisTex, KTH, KTH2, Drexel and UIUCTex) is much higher than that of object recognition datasets (MNIST, CIFAR-10 and DET). So, without explicit texture-feature extraction, a deep neural network cannot shatter the texture datasets because of their intrinsically high dimen-

sionality. However, as seen in Table 5.2, the features extracted from the texture datasets have a much lower intrinsic dimensionality and lie on a much lower dimensional manifold than the raw vectors and hence can be shattered/classified even by networks with relatively smaller architectures. Once, we have validated the fact that texture-based datasets lie on a higher dimensional manifold as compared to handwritten digit or object recognition datasets, we highlight issues associated with the high dimensionality of texture datasets.

5.5 Curse of Dimensionality in Texture Datasets

Curse of Dimensionality refers to the phenomenon where classification power of the model decreases with increase in dimensionality of the feature space of the input data. In the following sections, we derive some theoretical results on *Curse of Dimensionality* for high-dimensional texture data.

5.5.1 Sampling data in Higher Dimensional Manifolds

The mean distance from the centroid to the nearest data point is a useful metric for quantifying the hardness of classification [47]. To compute this mean distance, we first state a result on computing the expected value of a non-negative random variable and then use it to compute the mean distance from the centroid to the nearest sample point. The median distance was computed in [47]. However, to get a more accurate estimate of the distance metrics, we compute the mean in this chapter.

Lemma 5.5.1. *If a random variable y can take on only non-negative values, then the mean or expected value of y is given by $\int_0^\infty [1 - F_z(t)]dt$.*

Proof. Since $1 - F_Z(z) = P(Z \geq z) = \int_z^\infty f_Z(t)dt$, it follows that $\int_0^\infty (1 - F_Z(z)dz) = \int_0^\infty P(Z \geq z)dz = \int_0^\infty \int_z^\infty f_Z(t)dtdz$. Changing the order of integration, we have $\int_0^\infty (1 - F_Z(z)dz) = \int_0^\infty \int_0^t f_Z(t)dzdt = \int_0^\infty z[f_Z(t)]_0^t dt = \int_0^\infty t f_Z(t)dt$. Now, taking the substitution $t = z$ and $dt = dz$, the expected value

$$E(Z) = \int_0^\infty (1 - F_Z(z)dz) = \int_0^\infty (1 - y^p)^n dy \tag{5.18}$$

□

Lemma 5.5.2. *Consider n uniformly distributed samples in a unit hypersphere which is p-dimensional and centered at origin. Assuming an estimate of the nearest neighbor at the origin, the mean distance to the nearest data point from the origin is given by $\prod_{k=1}^{n}(1+\frac{1}{pk})^{-1}$.*

Proof. The volume of a ball of radius r in R^p is $\omega_p r^p$, where ω_p is given by $\frac{\pi^{p/2}}{(p/2)!}$ As a result, the probability that a point, taken uniformly in the unit ball, is within distance z of the origin is the volume of that ball divided by the volume of the unit ball. The factors of ω_p cancel, so we get the Cumulative Distribution Function (CDF) as $F(z) = z^p$, and Probability Density Function (PDF) as $f(z) = pz^{p-1}, 0 \leq z \leq 1$. From [50] we have the following general formula for the k^{th} order statistic of d points with CDF F and PDF f

$$g_k(l_k) = \frac{n!}{(k-1)!(n-k)!}[F(l_k)]^{k-1}[1-F(l_k)]^{d-k}f(l_k) \tag{5.19}$$

So, we have the minimum by setting $k = 1$ as

$$g(l) = d(1-F(l))^{(d-1)}f(l) = d(1-l^p)^{d-1}pl^{p-1} \tag{5.20}$$

This yields the CDF, $G(l) = 1-(1-l^p)^d$. The random variable l can take on only non-negative values. So, by Lemma 5.5.1 the mean or expected value is $E[Z] = \int_0^\infty [1-G_z(t)]dt$. Now, by substituting z^p by t, we have $E(Z) = \frac{1}{p}\int_0^1 t^{\frac{1}{p}-1}(1-t)^n dt$. (Note the change of limits since t lies in [0,1]).

This can be reduced using the Euler Gamma function as $E(Z) = \frac{1}{p}.\frac{\Gamma(\frac{1}{p})\Gamma(n+1)}{\Gamma(n+1+\frac{1}{p})}$. Now, by using the identity $\Gamma(t+1) = t\Gamma(t)$ recursively, we get Mean Distance,

$$D(p,N) = E(Z) = \prod_{k=1}^{n}(1+\frac{1}{pk})^{-1} \tag{5.21}$$

□

Table 5.3 shows the mean distance from the origin to the nearest sample point for various

Dataset	MNIST	CIFAR10	DET	Brodatz	VisTex	KTH	KTH2	Drexel	UIUC
D(p,N)	0.32	0.49	0.54	0.74	0.79	0.78	0.79	0.63	0.69

Table 5.3: Mean distance from origin to closest data point for various object recognition and texture datasets

datasets. From the table and according to [47], most data points for the texture datasets are nearer to the boundary of the feature space as compared to any other sample point. This makes prediction particularly difficult for these datasets because we cannot interpolate between data points and we need to extrapolate. Next, we propose a result on the expected distance from the origin to the farthest sample point and then use it to derive the relation of the *Relative Contrast* of the data points to the underlying dimensionality of the vector space as highlighted in Section 5.5.2.

Lemma 5.5.3. *Consider n uniformly distributed samples in a unit hypersphere which is p-dimensional and centered at origin. Assuming an estimate of the nearest neighbor at the origin, the mean distance to the farthest data point from the origin is given by* $1 - \frac{np}{(np+p-1)(np+p)}$.

Proof. Using equation 5.19, and setting $k = n$ for the maxima, we have,

$$g(l) = n[F(l)]^{n-1} f(l) = nl^{pn-1} pl^{p-1} = npl^{pn+p-2} \tag{5.22}$$

Therefore, the corresponding CDF is given by $G(l) = np\frac{l^{pn+p-1}}{pn+p-1}$. By Lemma 5.5.1, the mean or expected value is $E[X] = \int_0^\infty [1 - np\frac{l^{pn+p-1}}{pn+p-1}]dl$

$$= 1 - \frac{np}{(np+p-1)(np+p)} \tag{5.23}$$

□

Texture Datasets	**Brodatz**	**Drexel**	**KTH**	**KTH2**	**UIUCTex**	**VisTex**
CNN Test Error (%)	28.96	35.27	34.93	40.29	49.75	26.68

Table 5.4: Test Error of a Convolutional Neural Network trained using supervised backpropagation on the various texture datasets.

5.5.2 Relative Contrast in High Dimensions

In [14], it was shown that as dimensionality increases, the distance to the nearest neighbor tends to that of the farthest neighbor, i.e., contrast between points vanishes, while, in [2] it was shown that *Relative Contrast* varies as $\sqrt{p}$ for $n = 2$ sample points with dimensionality p. In this chapter, we generalize this to the case of n data points and also provide an exact estimate of the *Relative Contrast* instead of providing approximation bounds as [2]. We then show that as dimensionality $p \to \infty$, it yields the same result as [14] and [2]. Also, we eliminate the arbitrary constant C used in [2] which can vary significantly with change in parameters resulting in a fluctuating bound. It should be noted that we assume the L_2 norm distance metric and the Euclidean space for deriving our algebra.

Theorem 5.5.4. *If $RC_{n,p}$ be the Relative Contrast of n uniformly distributed sample points with p being the dimensionality of the underlying vector space, then, $RC_{n,p} = \frac{1-\frac{np}{(np+p-1)(np+p)}-\prod_{k=1}^{n}(1+\frac{1}{pk})^{-1}}{\prod_{k=1}^{n}(1+\frac{1}{pk})^{-1}}$ and $RC_{n,p}$ approaches 0 as p approaches ∞.*

Proof. From Lemma 5.5.2, we can see that the mean distance from origin to the nearest sampling point is denoted by the expression $\prod_{k=1}^{n}(1+\frac{1}{pk})^{-1}$. And from Lemma 5.5.3, the mean distance to the farthest sampling point is given by the expression $1-\frac{np}{(np+p-1)(np+p)}$. Therefore, $\frac{E[Dmax-Dmin]}{E[Dmin]}$

$$= \frac{1 - \frac{np}{(np+p-1)(np+p)} - \prod_{k=1}^{n}(1 + \frac{1}{pk})^{-1}}{\prod_{k=1}^{n}(1 + \frac{1}{pk})^{-1}} \tag{5.24}$$

Now, it can be easily shown that

$$\lim_{p\to\infty} \frac{1 - \frac{np}{(np+p-1)(np+p)} - \prod_{k=1}^{n}(1 + \frac{1}{pk})^{-1}}{\prod_{k=1}^{n}(1 + \frac{1}{pk})^{-1}} = 0 \tag{5.25}$$

Therefore, it follows that, $\frac{E[Dmax-Dmin]}{E[Dmin]} \to 0$ as $p \to \infty$. Therefore, $RC_{n,p} \to 0$ as $p \to \infty$. From equation 5.24, it can be concluded that for the general case of n sample points with a dimensionality of p, the expected value of the *relative contrast* for the sample points varies as $p^{-(n+1)}$. □

Theorem 5.5.5. *For $n = 2$ the general result proposed in Theorem 5.5.4 approaches the bound of $\frac{C}{\sqrt{p}}\sqrt{\frac{1}{2k+1}}$ proposed in [2] as the dimensionality p of the underlying sample space approaches ∞.*

Proof. From [2], it can be seen that for dimensionality of p and L_k norm,

$$\lim_{p\to\infty} E[\frac{Dmax - Dmin}{Dmin}.\sqrt{p}] = C\sqrt{\frac{1}{2k+1}} \tag{5.26}$$

Subtracting the rightmost term in Equation 5.24 from Equation 5.26, we have, $RC_{diff} = RC_{Agg} - RC_{Ours}$

$$= \frac{C}{\sqrt{p}}\sqrt{\frac{1}{2k+1}} - \frac{1 - \frac{np}{(np+p-1)(np+p)} - \prod_{k=1}^{n}(1+\frac{1}{pk})^{-1}}{\prod_{k=1}^{n}(1+\frac{1}{pk})^{-1}} \tag{5.27}$$

Therefore, for any arbitrary constant C and a given k,

$$lim_{p\to\infty}RC_{diff} = \lim_{p\to\infty}\Big(\frac{C}{\sqrt{p}}\sqrt{\frac{1}{2k+1}} - \frac{1 - \frac{np}{(np+p-1)(np+p)} - \prod_{k=1}^{n}(1+\frac{1}{pk})^{-1}}{\prod_{k=1}^{n}(1+\frac{1}{pk})^{-1}}\Big) \tag{5.28}$$

Therefore, by substituting $n = 2$ in Equation 5.28, it is easy to show that $lim_{p\to\infty}RC_{diff} = 0$. □

Theorem 5.5.4 validates the result in [14] and Theorem 5.5.5 shows that for the special case $n = 2$, our result approaches the bound of [2] as dimensionality p approaches ∞. So, from Section 5.4 and Theorem 5.5.4, we conclude that texture datasets lie on an inherently higher dimensional manifold than object recognition datasets, so the Relative Contrast of the texture datasets is lower.

5.6 Experiments

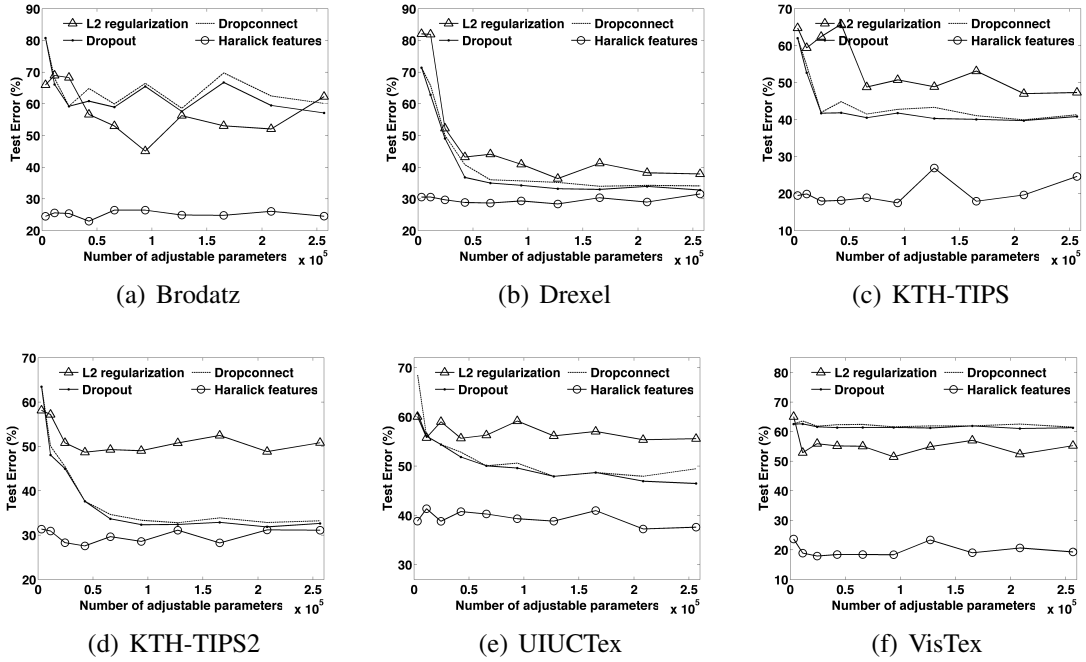
To validate our theory that error rate for networks with Haralick features is lower than that of raw vectors, we performed experiments on 6 benchmark texture classification datasets - Bro-

datz, VisTex, Drexel, KTH-TIPS, KTH-TIPS2 and UIUCTex. We extracted 27 features based on the GLCM metrics presented in Section 5.2. Without loss of generality, we select image size n^5 to be 28 and number of color levels k as 256. Also, datasets with multiple color channels are converted to grayscale. The Deep Neural Networks are trained by stacking – 1) Restricted Boltzmann Machines (RBM) and 2) Denoising Autoencoders (SDAE). Both the models are then discriminatively fine-tuned with supervised backpropagation. Figures 5.1 and 5.2 show the final test error of the backpropagation algorithm on the labeled test data using RBM and SDAE for unsupervised pre-training. Table 5.4 shows the final test error on the various texture datasets using a CNN. Our CNN has 3 convolutional layers with 32, 32 and 64 feature maps with 5×5 kernels each accompanied with max-pooling layers with 3×3 kernels. Each pooling layer is followed by a layer with Rectified Linear units and a local response normalization layer with a 3×3 locality. A softmax based loss function is used and the learning rate is initially set to 0.001 and then decreased as the inverse power of a gamma parameter (0.0001). In [44], the authors proposed a new CNN architecture for texture classification. However, in this chapter, we focus on the CNN architecture proposed in [67] to maintain uniformity with our theoretical analysis. By comparing the results in Figure 5.1, 5.2 and Table 5.4, we can see that for all texture datasets, Haralick feature based networks outperform the networks based on raw pixels. So, the experiments substantiate our theoretical claim that extraction of Haralick features create low-dimensional representations that enable Deep Neural Networks to achieve lower test error rate.

5.7 Discussion

The use of Deep Neural Networks for texture recognition has seen a significant impediment due to a lack of thorough understanding of the limitations of existing Neural architectures. In this chapter, we provide theoretical bounds on the use of Deep Neural Networks for texture data classification. First, using the theory of VC-dimension we establish the relevance of handcrafted feature extraction. As a corollary to this analysis, we derive for the first time upper bounds on the VC dimension of CNN as well as Dropout and Dropconnect networks and the relation between

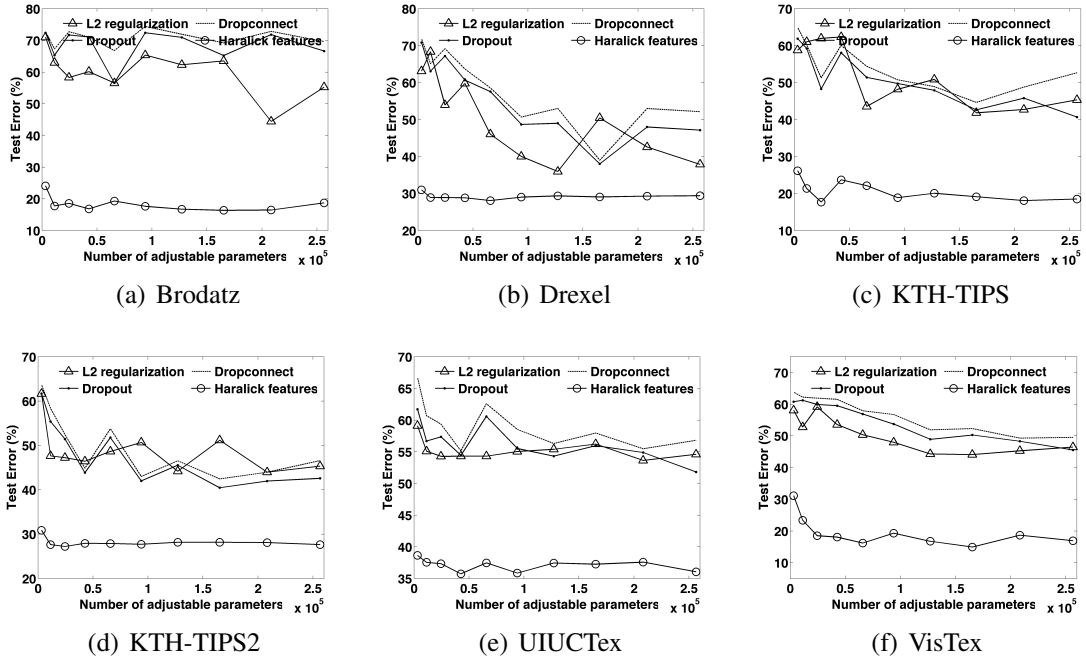
[5]Note that we extract $n\times n$ sliding window blocks from the various texture datasets for uniformity of analysis.



(a) Brodatz (b) Drexel (c) KTH-TIPS

(d) KTH-TIPS2 (e) UIUCTex (f) VisTex

Figure 5.1: Test Error on the 6 texture datasets with the Haralick features and stacked Restricted Boltzmann Machines with L_2 norm regularization, Dropout and Dropconnect obtained by varying the number of adjustable parameters.



(a) Brodatz (b) Drexel (c) KTH-TIPS

(d) KTH-TIPS2 (e) UIUCTex (f) VisTex

Figure 5.2: Test Error on the 6 texture datasets with the Haralick features and Stacked Denoising Autoencoders with L_2 norm regularization, Dropout and Dropconnect obtained by varying the number of adjustable parameters.

excess error rates. Then we use the concept of *Intrinsic Dimension* to show that texture datasets have a higher dimensionality than color/shape based data. Finally, we derive an important result on *Relative Contrast* that generalizes the one proposed in [2]. From the theoretical and empirical analysis, we conclude that for texture data, we need to redesign neural architectures and devise new learning algorithms that can learn GLCM or Haralick-like features from input data.

Chapter 6
Conclusions and Future Directions

Our probabilistic framework proposed in Chapter 2 has proved to be a useful tool for analyzing 1-m NAIP imagery for large-scale tree-cover mapping. Preliminary results on the NAIP tiles for California have produced positive detection rates of $\sim$86% for densely forested areas and $\sim$74% for urban areas. Comparative studies with NLCD show the effectiveness of our approach towards creating the NAIP 1-m "Golden Dataset". Validation with high-resolution airborne LiDAR data shows average positive detection rates of around 83% and average false positive rate as low as 10%. This proves the efficacy of our approach in creating high-resolution tree-cover maps for the entire country. The algorithm scales seamlessly to millions of scenes and can handle high variations, which is often the case for aerial imagery. The use of handcrafted features extracted from the Hue, Saturation, Intensity and NIR channels provides a useful framework for classifying NAIP imagery. The integration of the structured prediction framework based on Conditional Random Field helped to increase the True Positive Rates and decrease the False Positive Rate by incorporating classifier outputs from the neighboring pixels located within the same neighborhood system. The Near Infrared channel in the NAIP dataset was also useful in segregating regions with chlorophyll from others and proved to be a very useful discriminative feature for addressing the tree/non tree classification of the 1-m NAIP dataset.

The semi-supervised learning framework of Chapter 3 produces an accuracy of 97.95% and 93.9% on the SAT-4 and SAT-6 datasets and significantly outperforms the state-of-the-art by $\sim$11% and $\sim$15% respectively. The Feature extraction phase is inspired by the remote sensing literature and significantly improves the discriminative power of the framework. For satellite datasets, with inherently high variability, traditional deep learning approaches are unable to converge to a global optima even with significantly big and deep architectures. A statistical analysis based on Distribution Separability Criterion justifies the effectiveness of our feature extraction approach. The theoretical analysis based on Intrinsic Dimension and the subsequent derivation

of the mean distance from the centroid of the sample data to the nearest sampling point shows why prediction becomes difficult in higher dimensions.

Our learning framework based on probabilistic quadtrees significantly outperforms traditional Deep Belief Networks on both MNIST and n-MNIST datasets as well as Bangla and n-Bangla datasets. Probabilistic quadtrees help in generating sparse representations for the dataset and significantly improve the discriminative power of the framework.

Finally, the theoretical analysis of Deep Neural Networks applied to the problem of texture classification highlights the differences between texture datasets and other object recognition datasets. It also derives some useful results on the VC dimension of certain classes of Deep Neural Networks and showcases the difficulty of texture classification in terms of intrinsic dimensionality and relative contrast of the datasets.

The probabilistic framework proposed in Chapter 2 can be used to generate tree-cover maps for the entire continental United States. The framework can also be extended to more classes, such as different types of tree canopies, grasslands, croplands, etc., in the future.

It would be interesting to investigate the use of various pooling techniques like SPM [64] as well as certain sparse representations like sparse coding [68] and Hierarchical representations like Convolutional DBN [69] to handle satellite datasets. We believe that SAT-4 and SAT-6 will enable researchers to learn better representations for satellite datasets and create benchmarks for the classification of satellite imagery.

The framework with Probabilistic quadtrees presented in Chapter 4 can be extended to incorporate improved homogeneity criteria. Integration of the probabilistic quadtrees with other learning algorithms like Convolutional Neural Networks and Stacked Autoencoders provides another interesting direction for investigation.

Finally, since we have seen in the literature that for object recognition problems Convolutional Neural Networks (with a convolution and max-pooling kernel based architecture) usually outperform Deep Neural Networks, so as part of the theoretical analysis presented in Chapter 5, it would be useful to derive a relation between the VC dimensions of these networks and their er-

ror rates to provide a mathematical justification behind the performance gain of CNN over other deep neural architectures.

References

[1] Definiens AG. http://www.definiens.com, 2008.

[2] Charu C. Aggarwal, Alexander Hinneburg, and Daniel A. Keim. On the surprising behavior of distance metrics in high dimensional space. In *Lecture Notes in Computer Science*, pages 420–434. Springer, 2001.

[3] Huseyin Gokhan Akcay and Selim Aksoy. Automatic detection of geospatial objects using multiple hierarchical segmentations. *IEEE Transactions on Geoscience and Remote Sensing*, 46(7):2097–2111, 2008.

[4] R. Archibald and G. Fann. Feature selection and classification of hyperspectral images with support vector machines. *Geoscience and Remote Sensing Letters, IEEE*, 4(4):674–677, Oct 2007.

[5] Gükhan H. Bakir, Thomas Hofmann, Bernhard Schölkopf, Alexander J. Smola, Ben Taskar, and S. V. N. Vishwanathan. *Predicting Structured Data (Neural Information Processing)*. The MIT Press, 2007.

[6] Peter L. Bartlett and Wolfgang Maass. Vapnik-chervonenkis dimension of neural nets, 2003.

[7] Saikat Basu, S Ganguly, RR Nemani, S Mukhopadhyay, C Milesi, P Votava, A Michaelis, G Zhang, BD Cook, SS Saatchi, et al. A semi-automated machine learning algorithm for tree cover delineation from 1-m naip imagery using a high performance computing architecture. *AGU Fall Meeting Abstracts*, 1:3698, 2014.

[8] Saikat Basu, Sangram Ganguly, Andrew Michaelis, Petr Votava, Anshuman Roy, Supratik Mukhopadhyay, and Ramakrishna Nemani. A high performance computing approach to tree cover delineation in 1-m naip imagery using a probabilistic learning framework. 2015.

[9] Saikat Basu, Sangram Ganguly, Supratik Mukhopadhyay, Robert DiBiano, Manohar Karki, and Ramakrishna Nemani. Deepsat: A learning framework for satellite imagery. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '15, pages 37:1–37:10, New York, NY, USA, 2015. ACM.

[10] Saikat Basu, Sangram Ganguly, Ramakrishna R Nemani, Supratik Mukhopadhyay, Gong Zhang, Cristina Milesi, Andrew Michaelis, Petr Votava, Ralph Dubayah, Laura Duncanson, Bruce Cook, Yifan Yu, Saatchi Sassan, Robert DiBiano, Manohar Karki, Edward Boyda, Uttam Kumar, and Shuang Li. A semiautomated probabilistic framework for tree-cover delineation from 1-m naip imagery using a high-performance computing architecture. *Geoscience and Remote Sensing, IEEE Transactions on*, 53(10):5690–5708, 2015.

[11] Saikat Basu, Manohar Karki, Robert DiBiano, Supratik Mukhopadhyay, Sangram Ganguly, Ramakrishna Nemani, and Shreekant Gayaka. A theoretical analysis of deep neural networks for texture classification. In *Proceedings of the International Joint Conference on Neural Networks, IJCNN*, 2016.

[12] Saikat Basu, Manohar Karki, Sangram Ganguly, Robert DiBiano, Supratik Mukhopadhyay, and Ramakrishna Nemani. Learning sparse feature representations using probabilistic quadtrees and deep belief nets. In *Proceedings of the European Symposium on Artificial Neural Networks, ESANN*, 2015.

[13] Yoshua Bengio. Learning deep architectures for AI. *Found. Trends Mach. Learn.*, 2(1):1–127, January 2009.

[14] Kevin S. Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. When is "nearest neighbor" meaningful? In *Proceedings of the 7th International Conference on Database Theory*, ICDT '99, pages 217–235, London, UK, UK, 1999. Springer-Verlag.

[15] S. Bhagavathy and B. S. Manjunath. Modeling and detection of geospatial objects using texture motifs. *IEEE Transactions on Geoscience and Remote Sensing*, 44(12):3706–3715, Dec 2006.

[16] Ujjwal Bhattacharya and Bidyut B Chaudhuri. Handwritten numeral databases of indian scripts and multistage recognition of mixed numerals. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(3):444–457, 2009.

[17] Monica Bianchini and Franco Scarselli. On the complexity of neural network classifiers: A comparison between shallow and deep architectures. *IEEE Trans. Neural Netw. Learning Syst.*, 25(8):1553–1565, 2014.

[18] Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Inc., New York, NY, USA, 1995.

[19] Y-Lan Boureau, Jean Ponce, and Yann Lecun. A theoretical analysis of feature pooling in visual recognition. In *27th International Conference on Machine Learning, Haifa, Isreal*, 2010.

[20] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(11):1222–1239, November 2001.

[21] Leo Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, October 2001.

[22] R. Sabourin C. Freitas, L. S. Oliveira and F. Bortolozzi. Brazilian forensic letter database. In *11th International Workshop on Frontiers on Handwriting Recognition*, 2008.

[23] Miguel A. Carreira-Perpinan and Geoffrey E. Hinton. On contrastive divergence learning. In *AISTATS*, volume 10, pages 33–40, 2005.

[24] Claudio Ceruti, Simone Bassis, Alessandro Rozza, Gabriele Lombardi, Elena Casiraghi, and Paola Campadelli. Danco: An intrinsic dimensionality estimator exploiting angle and norm concentration. *Pattern Recognition*, 47(8):2569 – 2581, 2014.

[25] Daniel S. Chapman, Aletta Bonn, William E. Kunin, and Stephen J. Cornell. Random forest characterization of upland vegetation and management burning from aerial imagery. *Journal of Biogeography*, 37(1):37–46, 2010.

[26] Tse-Wei Chen, Yi-Ling Chen, and Shao-Yi Chien. Fast image segmentation based on k-means clustering with histograms in hsv color space. In *MMSP*, pages 322–325. IEEE Signal Processing Society, 2008.

[27] Dan Claudiu Ciresan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. In *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, CVPR '12, pages 3642–3649, Washington, DC, USA, 2012. IEEE Computer Society.

[28] David A. Clausi. An analysis of co-occurrence texture statistics as a function of grey level quantization. *Can. J. Remote Sensing*, 28(1):45–62, 2002.

[29] Bruce D. Cook, Lawrence A. Corp, Ross F. Nelson, Elizabeth M. Middleton, Douglas C. Morton, Joel T. McCorkel, Jeffrey G. Masek, Kenneth J. Ranson, Vuong Ly, and Paul M. Montesano. NASA Goddards LiDAR, Hyperspectral and Thermal (G-LiHT) Airborne Imager. *Remote Sensing*, 5(8):4045–4066, 2013.

[30] Y. Costa, L. Oliveira, A. Koerich, and F. Gouyon. Music genre recognition using gabor filters and lpq texture descriptors. In *Iberoamerican Congress on Pattern Recognition*, 2013.

[31] Jia Deng, Wei Dong, Richard Socher, Li jia Li, Kai Li, and Li Fei-fei. Imagenet: A large-scale hierarchical image database. In *In CVPR*, 2009.

[32] Kun Duan. Conditional random field models for structured visual object recognition, 2014.

[33] L.I. Duncanson, B.D. Cook, G.C. Hurtt, and R.O. Dubayah. An efficient, multi-layered crown delineation algorithm for mapping individual tree structure across multiple ecosystems. *Remote Sensing of Environment*, 154:378–386, Nov 2014.

[34] R. A. Finkel and J. L. Bentley. Quad trees a data structure for retrieval on composite keys. *Acta Informatica*, 4(1):1–9, 1974.

[35] Mark A. Friedl, Damien Sulla-Menashe, Bin Tan, Annemarie Schneider, Navin Ramankutty, Adam Sibley, and Xiaoman Huang. Modis collection 5 global land cover: Algorithm refinements and characterization of new datasets. *Remote Sensing of Environment*, 114:168–182, 2009.

[36] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36:193–202, 1980.

[37] S Ganguly, S Basu, S Mukhopadhyay, A Michaelis, C Milesi, P Votava, and RR Nemani. Deriving continuous fields of tree cover at 1-m over the continental united states from the national agriculture imagery program (naip) imagery to reduce uncertainties in forest carbon stock estimation. *AGU Fall Meeting Abstracts*, 1:08, 2013.

[38] Sangram Ganguly, RR Nemani, S Basu, S Mukhopadhyay, A Michaelis, and P Votava. Large-scale image analytics using deep learning. *AGU Fall Meeting Abstracts*, 1:3791, 2014.

[39] Pall Oskar Gislason, Jon Atli Benediktsson, and Johannes R. Sveinsson. Random forests for land cover classification. *Pattern Recognition Letters*, 27(4):294 – 300, 2006. Pattern Recognition in Remote Sensing (PRRS 2004).

[40] P.O. Gislason, J.A. Benediktsson, and J.R. Sveinsson. Random forest classification of multisource remote sensing and geographic data. In *Geoscience and Remote Sensing Symposium, 2004. IGARSS '04. Proceedings. 2004 IEEE International*, volume 2, pages 1049–1052 vol.2, Sept 2004.

[41] François A. Gougeon. A crown-following approach to the automatic delineation of individual tree crowns in high spatial resolution aerial images. *Canadian Journal of Remote Sensing*, 21(3):274–284, 1995.

[42] Leo Grady. Random walks for image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(11):1768–1783, November 2006.

[43] Dihua Guo, Hui Xiong, Vijayalakshmi Atluri, and NabilR. Adam. Object discovery in high-resolution remote sensing images: a semantic perspective. *Knowledge and Information Systems*, 19(2):211–233, 2009.

[44] Luiz Gustavo Hafemann. An analysis of deep neural networks for texture classification. 2014.

[45] R.K. Hanusiak, L.S. Oliveira, E. Justino, and R. Sabourin. Writer verification using texture-based features. *International Journal on Document Analysis and Recognition (IJDAR)*, 15(3):213–226, 2012.

[46] R. M. Haralick, K. Shanmugam, and Its'Hak Dinstein. Textural features for image classification. *Systems, Man and Cybernetics, IEEE Transactions on*, SMC-3(6):610–621, November 1973.

[47] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

[48] Geoffrey E. Hinton and Simon Osindero. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:2006, 2006.

[49] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012.

[50] Robert V. Hogg, Joseph W. McKean, and Allen T. Craig. *Introduction to mathematical statistics*. Pearson Education, 2005.

[51] Ashley C. Holt, Edmund Y. W. Seto, Tom Rivard, and Peng Gong. Object-based detection and classification of vehicles from highresolution aerial photography. *Photogrammetric Engineering and Remote Sensing*, 75(7):871–880, 2009.

[52] D. H. Hubel and T. N. Wiesel. Receptive fields, binocular interaction, and functional architecture in the cat's visual cortex. *Journal of Physiology (London)*, 160:106–154, 1962.

[53] A. Huete, K. Didan, T. Miura, E. P. Rodriguez, X. Gao, and L. G. Ferreira. Overview of the radiometric and biophysical performance of the MODIS vegetation indices. *Remote Sensing of Environment*, 83(1-2):195–213, November 2002.

[54] A Huete, K Didan, T Miura, E.P Rodriguez, X Gao, and L.G Ferreira. Overview of the radiometric and biophysical performance of the {MODIS} vegetation indices. *Remote Sensing of Environment*, 83(12):195 – 213, 2002. The Moderate Resolution Imaging Spectroradiometer (MODIS): a new generation of Land Surface Monitoring.

[55] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678. ACM, 2014.

[56] Y.J. Kaufman and D. Tanre. Atmospherically resistant vegetation index (ARVI) for EOS-MODIS. *Geoscience and Remote Sensing, IEEE Transactions on*, 30(2):261–270, Mar 1992.

[57] Leen kiat Soh and Costas Tsatsoulis. Texture analysis of SAR sea ice imagery using gray level co-occurrence matrices. *Geoscience and Remote Sensing, IEEE Transactions on*, pages 780–795, 1999.

[58] Pushmeet Kohli, L'Ubor Ladický, and Philip H. Torr. Robust higher order potentials for enforcing label consistency. *Int. J. Comput. Vision*, 82(3):302–324, May 2009.

[59] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. The MIT Press, 2009.

[60] R. Komura, Mamoru Kubo, and Ken-ichiro Muramoto. Delineation of tree crown in high resolution satellite image using circle expression and watershed algorithm. In *Geoscience and Remote Sensing Symposium, 2004. IGARSS '04. Proceedings. 2004 IEEE International*, volume 3, pages 1577–1580 vol.3, Sept 2004.

[61] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[62] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[63] S. Lazebnik, C. Schmid, and J. Ponce. A sparse texture representation using local affine regions. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(8):1265–1278, Aug 2005.

[64] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2*, CVPR '06, pages 2169–2178, Washington, DC, USA, 2006. IEEE Computer Society.

[65] Quoc V. Le, Marc'Aurelio Ranzato, Rajat Monga, Matthieu Devin, Greg Corrado, Kai Chen, Jeffrey Dean, and Andrew Y. Ng. Building high-level features using large scale unsupervised learning. In *ICML*, 2012.

[66] Donald G. Leckie, Franois A. Gougeon, Nicholas Walsworth, and Dennis Paradine. Stand delineation and composition estimation using semi-automated individual tree crown analysis. *Remote Sensing of Environment*, 85(3):355 – 369, 2003.

[67] Yann Lecun, Lon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998.

[68] Honglak Lee, Alexis Battle, Rajat Raina, and Andrew Y. Ng. Efficient sparse coding algorithms. In *In NIPS*, pages 801–808. NIPS, 2007.

[69] Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 609–616, New York, NY, USA, 2009. ACM.

[70] C. Leistner, H. Grabner, and H. Bischof. Semi-supervised boosting using visual similarity learning. In *IEEE Conference on Computer Vision and Pattern Recognition, 2008. CVPR 2008.*, pages 1–8, June 2008.

[71] Elizaveta Levina and Peter J. Bickel. Maximum Likelihood Estimation of Intrinsic Dimension. In *NIPS*, 2004.

[72] J. Li, R. Nevatia, and S. Nornoha. User assisted modeling of buildings from aerial images. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1999.*, volume 2, pages –279 Vol. 2, 1999.

[73] A. Liaw and M. Weiner. Classification and regression by random forests. *R News*, 2(3):18–22, 2002.

[74] G.C. Liknes, C.H. Perry, and D.M. Meneguzzo. Assessing tree cover in agricultural landscapes using high resolution aerial imagery. *J. Terres. Obs*, 2:38–55, 2010.

[75] J. Makhoul, R. Schwartz, and A. El-Jaroudi. Classification capabilities of two-layer neural nets. In *Acoustics, Speech, and Signal Processing, 1989. ICASSP-89., 1989 International Conference on*, pages 635–638 vol.1, May 1989.

[76] Volodymyr Mnih and Geoffrey Hinton. Learning to detect roads in high-resolution aerial images. In *Proceedings of the 11th European Conference on Computer Vision (ECCV)*, September 2010.

[77] Abdel-rahman Mohamed, George E. Dahl, and Geoffrey E. Hinton. Acoustic modeling using deep belief networks. *IEEE Transactions on Audio, Speech & Language Processing*, 20(1):14–22, 2012.

[78] Derrick Nguyen and Bernard Widrow. Improving the learning speed of 2-layer neural networks by choosing. In *Initial Values of the Adaptive Weights, International Joint Conference of Neural Networks*, pages 21–26, 1990.

[79] Richard Nock and Frank Nielsen. Statistical region merging. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 26:1452–1458, 2004.

[80] Geoffrey Oxholm, Prabin Bariya, and Ko Nishino. The scale of geometric texture. In *Computer Vision ECCV 2012*, volume 7572 of *LNCS*, pages 58–71. Springer Berlin Heidelberg, 2012.

[81] L. S. Oliveira P. L. de Paula Filho and A. S. Britto J. A database for forest species recognitio. In *In Proceedings of the XXII Brazilian Symposium on Computer Graphics and Image Processing*, 2009.

[82] D. A. Pouliot, D. J. King, F. W. Bell, and D. G. Pitt. Automated tree crown detection and delineation in high-resolution digital camera imagery of coniferous forest regeneration. *Remote Sens. Environ.*, 82(2-3):322–334, 2002.

[83] Marc'Aurelio Ranzato, Y-lan Boureau, and Yann Lecun. Sparse Feature Learning for Deep Belief Networks. In *Advances in Neural Information Processing Systems*, 2008.

[84] Adriana Romero, Carlo Gatta, and Gustavo Camps-Valls. Unsupervised deep feature extraction of hyperspectral images. 2014.

[85] J. W. Rouse, R. H. Haas, J. A. Schell, and D. W. Deering. Monitoring vegetation systems in the great plains with ERTS. *NASA Goddard Space Flight Center 3d ERTS-1 Symposium*, pages 309–317, 1974.

[86] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, pages 1–42, April 2015.

[87] Robert E. Schapire. The strength of weak learnability. *Mach. Learn.*, 5(2):197–227, July 1990.

[88] Eitan Sharon, Achi Brandt, and Ronen Basri. Segmentation and boundary detection using multiscale intensity measurements. In *CVPR (1)*, pages 469–476. IEEE Computer Society, 2001.

[89] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:888–905, 1997.

[90] Carlos N. Silla Jr., Alessandro L. Koerich, and Celso A. A. Kaestner. The latin music database. In *Proceedings of the 9th International Conference on Music Information Retrieval*, pages 451–456, Philadelphia, USA, September 14-18 2008. `http://ismir2008.ismir.net/papers/ISMIR2008_106.pdf`.

[91] Xian Sun, Hongqi Wang, and Kun Fu. Automatic detection of geospatial objects using taxonomic semantics. *Geoscience and Remote Sensing Letters, IEEE*, 7(1):23–27, Jan 2010.

[92] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. DeepFace: Closing the Gap to Human-Level Performance in Face Verification. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[93] P. Tokarczyk, J. Montoya, and K. Schindler. An evaluation of feature learning methods for high resolution image classification. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, I-3:389–394, 2012.

[94] Zhuowen Tu, Xiangrong Chen, Alan L. Yuille, and Song chun Zhu. Image parsing: Unifying segmentation, detection, and recognition. In *International Journal of Computer Vision*, volume 63, pages 18–25, 2003.

[95] Compton J. Tucker. Red and photographic infrared linear combinations for monitoring vegetation. *Remote Sensing of Environment*, 8(2):127 – 150, 1979.

[96] C. Vaduva, I. Gavat, and M. Datcu. Deep learning in very high resolution remote sensing image information mining communication concept. In *Signal Processing Conference (EUSIPCO), 2012 Proceedings of the 20th European*, pages 2506–2510, Aug 2012.

[97] V. N. Vapnik and A. Ya. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2):264–280, 1971.

[98] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.*, 11:3371–3408, December 2010.

[99] J.E. Vogelmann, T.L. Sohl, P.V. Campbell, and D.M. Shaw. Regional land cover characterization using Landsat Thematic Mapper Data and Ancillary Data Sources. *Environmental Monitoring and Assessment*, 51(1-2):415–428, 1998.

[100] James D. Wickham, Stephen V. Stehman, Leila Gass, Jon Dewitz, Joyce A. Fry, and Timothy G. Wade. Accuracy assessment of NLCD 2006 land cover and impervious surface. *Remote Sensing of Environment*, 130:294–304, 2013.

[101] WWW. Brodatz. http://sipi.usc.edu/database/database.php?volume=textures.

[102] WWW. Datasets. https://drive.google.com/uc?id=0B0Fef71_vt3PUkZ4YVZ5WWNvZWs&export=download.

[103] WWW. KTH. http://www.nada.kth.se/cvap/databases/kth-tips/index.html.

[104] WWW. Landsat. http://landsat.usgs.gov/.

[105] WWW. Mnist. http://yann.lecun.com/exdb/mnist/.

[106] WWW. Modis. http://vip.arizona.edu/documents/MODIS/MODIS_VI_UsersGuide_01_2012.pdf.

[107] WWW. n-mnist. http://csc.lsu.edu/~saikat/n-mnist/.

[108] WWW. Naip. http://www.fsa.usda.gov/Internet/FSA_File/naip_2009_info_final.pdf.

[109] WWW. Nlcd. http://www.gsd.harvard.edu/gis/manual/earthshelter/National%20Land-Cover%20Dataset%20%28NLCD%29%20Metadata%20%20US%20EPA.htm.

[110] WWW. Noisy bangla. http://csc.lsu.edu/~saikat/noisy-bangla/.

[111] WWW. Pleiades. https://nex.nasa.gov/nex/static/media/other/Intro_to_Pleiades.pdf.

[112] WWW. R-project. http://www.r-project.org.

[113] WWW. Tree cover presentation svalbard. http://www.nina.no/Portals/NINA/Bilder%20og%20dokumenter/Prosjekter/ArcticBiomass/Ganguly_Svalbard.pdf.

[114] WWW. VisTex. http://vismod.media.mit.edu/vismod/imagery/VisionTexture/vistex.html.

[115] Gong Zhang, S Ganguly, RR Nemani, C Milesi, S Basu, and U Kumar. Reducing uncertainties in satellite-derived forest aboveground biomass estimates using a high resolution forest cover map. *AGU Fall Meeting Abstracts*, 1:0199, 2014.

Chapter 7
Appendix A: Permission to reprint from ACM

ASSOCIATION FOR COMPUTING MACHINERY, INC. LICENSE TERMS AND CONDITIONS

May 18, 2016

This is a License Agreement between Saikat Basu (”You”) and Association for Computing Machinery, Inc. (”Association for Computing Machinery, Inc.”) provided by Copyright Clearance Center (”CCC”). The license consists of your order details, the terms and conditions provided by Association for Computing Machinery, Inc., and the payment terms and conditions.

License Number 3872150175516

License date May 18, 2016

Licensed content publisher Association for Computing Machinery, Inc.

Licensed content publication Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems

Licensed content title DeepSat: a learning framework for satellite imagery

Licensed content author Saikat Basu, et al

Licensed content date Nov 3, 2015

Type of Use Thesis/Dissertation

Requestor type Author of this ACM article

Is reuse in the author’s own new work? Yes

Format Print and electronic

Portion Full article

Will you be translating? No

Order reference number None

Title of your thesis/dissertation PROBABILISTIC AND DEEP LEARNING ALGORITHMS FOR THE ANALYSIS OF IMAGERY DATA

Expected completion date Dec 2016

Estimated size (pages) 130

Total 8.00 USD

Terms and Conditions

Rightslink Terms and Conditions for ACM Material

1. The publisher of this copyrighted material is Association for Computing Machinery, Inc. (ACM). By clicking ”accept” in connection with completing this licensing transaction, you agree that the following terms and conditions apply to this transaction (along with the Billing and Payment terms and conditions established by Copyright Clearance Center, Inc. (”CCC”), at the time that you opened your Rightslink account and that are available at any time at).

2. ACM reserves all rights not specifically granted in the combination of (i) the license details provided by you and accepted in the course of this licensing transaction, (ii) these terms and conditions and (iii) CCC’s Billing and Payment terms and conditions.

3. ACM hereby grants to licensee a non-exclusive license to use or republish this ACM-

copyrighted material* in secondary works (especially for commercial distribution) with the stipulation that consent of the lead author has been obtained independently. Unless otherwise stipulated in a license, grants are for one-time use in a single edition of the work, only with a maximum distribution equal to the number that you identified in the licensing process. Any additional form of republication must be specified according to the terms included at the time of licensing.

*Please note that ACM cannot grant republication or distribution licenses for embedded third-party material. You must confirm the ownership of figures, drawings and artwork prior to use.

4. Any form of republication or redistribution must be used within 180 days from the date stated on the license and any electronic posting is limited to a period of six months unless an extended term is selected during the licensing process. Separate subsidiary and subsequent republication licenses must be purchased to redistribute copyrighted material on an extranet. These licenses may be exercised anywhere in the world.

5. Licensee may not alter or modify the material in any manner (except that you may use, within the scope of the license granted, one or more excerpts from the copyrighted material, provided that the process of excerpting does not alter the meaning of the material or in any way reflect negatively on the publisher or any writer of the material).

6. Licensee must include the following copyright and permission notice in connection with any reproduction of the licensed material: "[Citation] YEAR Association for Computing Machinery, Inc. Reprinted by permission." Include the article DOI as a link to the definitive version in the ACM Digital Library. Example: Charles, L. "How to Improve Digital Rights Management," Communications of the ACM, Vol. 51:12, 2008 ACM, Inc.
http://doi.acm.org/10.1145/nnnnnn.nnnnnn (where nnnnnn.nnnnnn is replaced by the actual number).

7. Translation of the material in any language requires an explicit license identified during the licensing process. Due to the error-prone nature of language translations, Licensee must include the following copyright and permission notice and disclaimer in connection with any reproduction of the licensed material in translation: "This translation is a derivative of ACM-copyrighted material. ACM did not prepare this translation and does not guarantee that it is an accurate copy of the originally published work. The original intellectual property contained in this work remains the property of ACM."

8. You may exercise the rights licensed immediately upon issuance of the license at the end of the licensing transaction, provided that you have disclosed complete and accurate details of your proposed use. No license is finally effective unless and until full payment is received from you (either by CCC or ACM) as provided in CCC's Billing and Payment terms and conditions.

9. If full payment is not received within 90 days from the grant of license transaction, then any license preliminarily granted shall be deemed automatically revoked and shall be void as if never granted. Further, in the event that you breach any of these terms and conditions or any of CCC's Billing and Payment terms and conditions, the license is automatically revoked and shall be void as if never granted.

10. Use of materials as described in a revoked license, as well as any use of the materials beyond the scope of an unrevoked license, may constitute copyright infringement and publisher reserves the right to take any and all action to protect its copyright in the materials.

11. ACM makes no representations or warranties with respect to the licensed material and adopts on its own behalf the limitations and disclaimers established by CCC on its behalf in its

Chapter 8
Appendix B: Permission to reprint from IEEE

Title: A Semiautomated Probabilistic Framework for Tree-Cover Delineation From 1-m NAIP Imagery Using a High-Performance Computing Architecture

Author: Saikat Basu; Sangram Ganguly; Ramakrishna R. Nemani; Supratik Mukhopadhyay; Gong Zhang; Cristina Milesi; Andrew Michaelis; Petr Votava; Ralph Dubayah; Laura Duncanson; Bruce Cook; Yifan Yu; Sassan Saatchi; Robert DiBiano; Manohar Karki; Edward Boyda; Uttam Kumar; Shuang Li

Publication: Geoscience and Remote Sensing, IEEE Transactions on

Publisher: IEEE

Date: Oct. 2015

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license.

Vita

Saikat Basu completed his Bachelor of Technology in Computer Science and Engineering from National Institute of Technology, Durgapur in 2011. During his doctoral program, he has been doing research on the analysis of various kinds of imagery data using Computer Vision and Deep Learning algorithms. During his PhD, he has worked as an intern at NASA Ames Research Center, California and the Facebook Maps team in Boston. After graduating, he is going to join the Facebook Maps team as a Research Scientist.